

Colorado State University ECE Senior Design 2015		Team Members: Jared Romano Rachael Dinger Chris Jones Miles Kelly
Supervising Professor: Dr. George Collins	Cyber Security Senior Design Team	Industry Advisor: Dr. Joel Dubow
Implementing a Honeypot and Intrusion Detection System on a Raspberry Pi 2 Module		

Project Summary:

The Cyber Security Team's mission is to research and analyze strategies for triangulating the source of malicious cyber threats through the use of honeypot and intrusion detection system software and data analysis techniques. Our goal is to provide an encompassing viewpoint of the state of reactive cyber security techniques currently in use, and to provide insight on malware behavior and their intended targets.

Why is this Project Important?

As the technological capabilities of our society increases, the alluring real-time environment offered by the internet has increased the movement of commerce, media, education, and government institutions into the cyber world. Alas, the pilgrimage to enhanced efficiency and data access does not come without cost; as the number of cyber-crimes committed daily has never been higher, the need for enhanced cyber security has become precedent.

Date	Revision	Details	Approved
4/22/2015	1	Initial Revision	RD
4/29/2015	2	Additional Appendix Material Added	RD
4/30/2015	3	Comparison to Windows Honeypot Software & Budget Revision	GC

Abstract

Cyber security is a rapidly developing field that is constantly evolving to prevent newly emerging threats. Two such tools to help the fight against malicious attackers are intrusion detection systems (IDS) and honeypots. An intrusion detection system monitors network traffic and alerts the user if any suspicious data-packets are found; while honeypots emulate machines that hackers want to attack, with the goal of deflecting attacks and researching malicious intent.

The Cyber Security Team developed a network penetration testing environment during the spring semester to allow for honeypot testing at the Colorado State University. This desktop computer was specifically designed to have a lot of available RAM and processing power to be able to handle hosting multiple virtual machines at once. In addition, a portable honeypot solution, marketed toward a small business consumers, was created which contains an intrusion detection system and honeypot software.

Portability was achieved by installing Snort (intrusion detection system software) and Kippo (honeypot software) on a Raspberry Pi 2 module. The Raspberry Pi's small form factor, low cost, and decent processing power were the reasons why the Raspberry Pi 2 was chosen as the target platform. Aside from Snort, supporting software for Snort had to be installed – namely Barnyard2 and PulledPork. Barnyard2 imports activities logged by Snort to a MySQL database and PulledPork updates the security definitions utilized by Snort to inspect data-packets.

Additionally, supporting graphical software had to be installed to allow for the user-friendly visualization of the statistics gathered from both Snort and Kippo. A Ruby on Rails webserver was used as the interface between the different frontends. Additional third party software was also utilized as front-ends: for Kippo visualization, Kippo-Graphs was used to visualize statistics. For Snort, Aanval Tactical Flex, Inc. was utilized.

Performance of the Raspberry Pi 2, once loaded with an interface, was found to be decent. However, the team finds it questionable as to whether this module could handle more honeypot software due to restricted memory availability. Because of this, the team has made a series of future recommendations concerning both the development board and system architecture which are aimed at improving system viability.

Table of Contents

Abstract.....	2
List of Tables	7
Introduction.....	8
Scope.....	8
Creating the Cyber Security Lab’s Desktop Computer.....	8
Hardware Components Utilized.....	9
Software Components Installed	9
System Settings.....	10
Selecting the Raspberry Pi 2 Module for System Development.....	10
Advantages.....	11
Disadvantages	12
Third Party Software Installation & Interfaces	13
Third Party Software Installed	13
Configuration Information	14
Configuring Snort, Barnyard2 & PulledPork.....	14
Configuring Apache2, Kippo & Kippo-Graphs	17
Configuring Ruby on Rails	19
Database Implementation.....	19
Configuring the MySQL Database	19
Cerburi Database Schema	20
Ruby on Rails Interface: The Setup-Up Wizard and Dashboard	23
Background Information: Electing to Work with Ruby on Rails	23
The Set-Up Wizard	25
Username & Password (“Index”) Form.....	26
Oink-Code Form.....	27
Dashboard Form.....	32
Detail Pages	34
Snort Event Detail.....	34
Kippo Event Detail	36
Data Packet Detail Pages	37
Help/FAQ Page.....	40
Kippo-Graph Pages.....	40
Aanval by Tactical Flex, Inc.	43
Configuring Aanval	43

Aanval Homepage.....	43
Future Recommendations for Using Aanval.....	45
Database Integration for Rendering Charts.....	45
Proof of Concept: Testing the Cyber Cerburi System	47
Testing Configuration	47
Port Scanning.....	47
Procedure	48
Results & Discussion	49
Brute Force Attacks	49
Procedure	50
Results & Discussion	51
Proof of Concept: Conclusion.....	54
The Cyber Cerburi System vs. Window's Honeypots	54
Known Issues & Recommendations for System Improvement	56
Future Development Board Recommendations	57
Conclusion	59
References.....	59
Appendix A: In-House Testing Exploring Honeypots: An Instruction into Ethical Hacking and Intrusion Detection: Findings & Recommendations	60
About the Honeypot Lab.....	60
Findings	60
Recommendations.....	60
Appendix B: Abbreviations	61
Appendix C: Summary of Team Spending.....	62
Acknowledgements.....	63

List of Figures

Figure 1: Ruby on Rails Structure.....	24
Figure 2: Set-up Wizard Page	25
Figure 3: Application Layout, Referencing a Style Sheet.....	26
Figure 4: User Password Validation	26
Figure 5: Validating Password Requirements.....	27
Figure 6: Storing a Username & Password.....	27
Figure 7: Invoking Assets (Images)	28
Figure 8: Instructing Users on How to Obtain an Oink-Code: Steps 1 & 2.....	28
Figure 9: Instructing Users on How to Obtain an Oink-Code: Step 3	29
Figure 10: Instructing Users on How to Obtain an Oink-Code: Step 4	29
Figure 11: Instructing Users on How to Obtain an Oink-Code: Step 5	29
Figure 12: Instructing Users on How to Obtain an Oink-Code: Step 6	30
Figure 13: Locating Where to Place the Oink-Code.....	30
Figure 14: The Routes File	31
Figure 15: Cyber Cerburi Dashboard Page	32
Figure 16: Cyber Cerburi Dashboard, First Row of Charts	32
Figure 17: Cyber Cerburi Dashboard, 2nd Row of Charts.....	33
Figure 18: Cyber Cerburi Dashboard, 3rd Row of Charts	33
Figure 19: Cyber Cerburi Dashboard Page - Kippo Activity.....	33
Figure 20: Cyber Cerburi Dashboard, Information Provided in Sidebar	34
Figure 21: Snort Event Detail Page.....	35
Figure 22: Snort Event Detail - Data Table	35
Figure 23: Kippo Detail Page: Overview.....	36
Figure 24: Kippo Bubble Chart.....	36
Figure 25: Detailed Successful Session Information	37
Figure 26: Informational Sidebar Material	37
Figure 27: Ports Targeted for Attack Detail Page.....	38
Figure 28: Ports Targeted for Attack Bubble Chart.....	38
Figure 29: Ports Targeted for Attack Data Table.....	38
Figure 30: Malicious Source Port Detail.....	39
Figure 31: Sidebar Material for Both Pages.....	39
Figure 32: Cyber Cerburi Help/FAQ Page	40
Figure 33: Kippo-Graph Pages	41
Figure 34: Kippo-Graph Pages	41
Figure 35: Kippo-Graph Pages	42
Figure 36: Kippo-Graph Pages	42
Figure 37: Navigating to Aanval's Homepage	43
Figure 38: Aanval Log-In Page.....	43
Figure 39: Aanval Homepage	44
Figure 40: Aanval's Event Timeline Browser	44
Figure 41: Aanval's Frequent Offender Page.....	45
Figure 42: Embedded Ruby Code in the HTML Page.....	46
Figure 43: Initialization of the Model	46
Figure 44: Creating the Active Base Record	47
Figure 45: Locating the Cyber Cerburi System on a Network.....	48

Figure 46: Finding Open Ports.....	48
Figure 47: Recorded Port Scanning Activity by Cyber Cerburi	49
Figure 48: Recorded Port Scanning Activity by Cyber Cerburi	49
Figure 49: Recorded Port Scanning Activity by Aanval.....	49
Figure 50: Nmap Scan of the Cyber Cerburi System.....	50
Figure 51: Brute Force Attack	51
Figure 52: Performing a "wget" Command during a Brute-Force Attack	51
Figure 53: Capturing and Storing Downloaded Malware	52
Figure 54: "wget" Command Recorded by Kippo	52
Figure 55: Geolocation Information Provided	52
Figure 56: The Top SSH Clients Recorded by Kippo (Available in Kippo-Graphs)	53
Figure 57: Commonly Used Usernames (Reported by Cyber Cerburi).....	53
Figure 58: Command Line Hacking Inputs (Recorded by Cyber Cerburi).....	53

List of Tables

Table 1: Cyber Security Lab Desktop Computer - Hardware Components.....	9
Table 2: Host Machine Software Install List	9
Table 3: Available Virtual Machines	10
Table 4: System Settings (Usernames & Passwords)	10
Table 5: Advantages of Raspberry Pi 2	11
Table 6: Disadvantages of Raspberry Pi 2	12
Table 7: 3rd Party Software Install List	14
Table 8: Snort Configuration	15
Table 9: Barnyard2 Configuration	16
Table 10: PulledPork Configuration	16
Table 11: Apache2 Configuration.....	17
Table 12: Kippo Honeypot Configuration	18
Table 13: Kippo-Graph Configuration.....	19
Table 14: Important Ruby & Ruby on Rails Commands.....	19
Table 15: MySQL Database Configuration: Usernames & Passwords.....	20
Table 16: Schema for the 'events' Table, for the Associated 'Event' Model	21
Table 17: Schema for the 'kippo_auths' Table, for the Associated 'KippoAuth' Model	21
Table 18: Schema for the 'kippo_inputs' Table, for the Associated 'KippoInput' Model.....	21
Table 19: Schema for the 'kippo_sessions' Table, for the Associated KippoSession Model.....	22
Table 20: Schema for the 'sig_classes' Table, for the Associated SigClass Model.....	22
Table 21: Schema for the 'signatures' Table, for the Associated Signature Model.....	22
Table 22: Schema for the 'tcphdrs' Table, for the Associated Tcphdr Model	23
Table 23: Schema for the 'users' Table, for the Associated User Model.....	23
Table 24: Known Issues & Recommendations for System Improvement	57
Table 25: Development Board Pros & Cons.....	59
Table 26: Finalized Spending Report.....	62

Introduction

In the United States, the classic American dream has always involved an individual's ability to be self-employed, and to have ultimate control over one's success. From 1993 to 2009, this dream has held true for many Americans, with small firms contributing well over nine-million of the estimated 15 million total jobs created [1]. However, there is always a dark side to be considered: the rise in cybercrimes committed annually has continued to grow since the advent of the internet – and with this, a new pattern has emerged. Today, over 36% [1] of the cybercrimes committed are specifically aimed at taking down small businesses; at the end of 2011, this number was only 18% [1].

The fact that smaller businesses are becoming a more popular target for hackers is indeed, very unfortunate. What is even more alarming is that over 77% [1] of the small businesses polled in a recent survey claim that they are "...safe from cyber threats such as hackers, viruses, malware, or other cyber-security breaches [1]". An additional 18% [1] of those polled also alluded to the issue of basic network security ignorance, stating "...they would not know if their computer or network was compromised [1]".

Recognizing the need for more accessible network security options for vulnerable small business owners, the Cyber Security Senior Design Team has created a "honeypot in a pi": a plug-and-play honeypot system on a portable, efficient Raspberry Pi module. This system, deemed "Cyber Cerburi" is aimed at implementing advanced network security and monitoring techniques with minimal information supplied by the user. With over 60% [1] of targeted small businesses forced to close their doors as a direct result of a cybercrime, the goal for the team was very clear: deliver a system that is inexpensive, easy to use, and above all, detects and stops a hacker.

Scope

The scope of this document entails the work of the Cyber Security Senior Design Team, in its entirety, for the duration of the spring 2015 semester. Development board selection, third-party software selection, and selection of the development environment/language will be discussed. In addition, general information regarding installation and configuration of third party software, as well as a description of the graphical user interface system created will be disclosed.

System testing and analysis has also been performed as "proof of concept" of the Cyber Cerburi interface. Finally, further recommendations and possible marketing approaches are included for consideration for the continuation project that will extend into the scope of next year's Cyber Security Design Team.

Creating the Cyber Security Lab's Desktop Computer

The Cyber Security desktop computer's specifications were designated with the creation of the ultimate network penetration testing environment in mind. The team's main goal was to build both a fast and reliable computer that would allow multiple virtual machines to be hosted within a virtual network.

Special measures were also taken to image each virtual machine, as well as the hard drive of the computer. These tasks were carried out so that future teams could easily pick up where the 2014/2015 Cyber Security Team left off and also be able to recover in the event that a virtual machine, or the desktop's operating system becomes corrupt.

All additional parts and manuals have been set aside in secured storage, and will be provided to future teams for any modifications they may wish to perform.

Hardware Components Utilized

Table 1: Cyber Security Lab Desktop Computer - Hardware Components lists all of the hardware components purchased and installed by the team.

Component	Description
CPU	• AMD FX-8350 Black Edition Vishera • Core Count: 8-Core • Normal Operating Frequency: 4.0GHz
RAM	• G.SKILL Ripjaws X Series • Capacity: 32GB (4 x 8GB) • Type: 240-Pin DDR3 SDRAM • Speed: DDR3 1600 (PC3 12800)
Motherboard	• ASUS M5A99X EVO R2.0 • Socket: AM3+ • Peripherals: 6 x SATA 6Gb/s, USB 3.0 • Form Factor: ATX
Storage	• WD BLACK SERIES WD1003FZEX • Capacity: 1TB • RPM: 7200 RPM • Cache: 64MB
Power Supply	• CORSAIR CX series CX500 • Max power: 500W • Certification: Bronze
Wireless	• ASUS (USB-N10) Wireless-N USB Adapter • Speed: 150Mbps Transmit / 150Mbps Receive

Table 1: Cyber Security Lab Desktop Computer - Hardware Components

Software Components Installed

Below, *Table 2: Host Machine Software Install List*, displays all of the software packages loaded onto the Cyber Security Desktop computer's C: partition.

Component	Description
OS	Windows 7 x64-bit Edition – with Service Pack 1
PDF Reader	Foxit PDF Reader, with PDF Printing
Virtual Machine	VMWare Player
Anti-Virus	AVG Free Edition

Table 2: Host Machine Software Install List

The D: (Data) partition contains all virtual machines, technical documents, and Raspberry Pi images associated with the project, while the R: (Recovery) partition contains all necessary Windows drivers as additional back-up software.

Table 3: Available Virtual Machines, displayed below, lists all of the virtual machines that have been built/configured and loaded onto the Cyber Security Desktop computer by the team.

VM Name	Operating System Type	Additional Software Installed
Grey Hat	Kali Linux x86 (32-bit Operating System)	Kali Linux bundle, Snort, VMWare tools
Grey Hat 64	Kali Linux x64 (64-bit Operating System)	Kali Linux bundle, VMWare tools
HoneyDrive	X-Ubuntu 12.4.4	HoneyDrive Honeypot bundle
HP	Windows XP – Professional Edition, no software updates	Wireshark, KF Sensor Honeypot, VMWare tools
HP updated	Windows XP – Professional Edition, with software updates	Wireshark, KF Sensor Honeypot, Firefox Web Browser, Autopsy Forensic Tool, VMWare tools
VS	Windows 7	Wireshark, VMWare tools
HP DEV	Ubuntu14 x 86 (32-bit Operating System)	Snort, Kippo, Barnyard2, Kippo-Graphs, Apache2, MySQL, VMWare tools, SSH, Python-Twisted, Php5, Perl,

Table 3: Available Virtual Machines

System Settings

All usernames and passwords for the host machine, as well as the assembled virtual machines are displayed below in *Table 4: System Settings (Usernames & Passwords)*.

Machine	User Name	Password
Host Machine	Cyber Security	N/A
Grey Hat	root	toor
Grey Hat 64	root	toor
HoneyDrive	honeydrive	honeydrive
HP	Admin	N/A
HP updated	Admin	N/A
VS	Admin	N/A
HP DEV	comsec	password

Table 4: System Settings (Usernames & Passwords)

Selecting the Raspberry Pi 2 Module for System Development

The Cyber Security Team had to keep a close eye on the budget and form factor (for portability) when making decisions concerning the target hardware setup. The team considered two development boards as possible candidates: the Arduino board and the Raspberry Pi 2. The Arduino development board was quickly ruled out, as the technical specifications do not fit well into the project.

After some consideration, and with the coinciding release of the Raspberry Pi 2 module, the Raspberry Pi was chosen as the team's development board. Below are some advantages and disadvantages of the Raspberry Pi 2 as a development board for this project.

Advantages

Some of the reasons why the Cyber Security Team chose to work with the Raspberry Pi 2 module are displayed in Table 5: Advantages of Raspberry Pi 2 below.

Advantages of Choosing the Raspberry Pi 2 Module	
Cost – Very Inexpensive to Implement	• Development board is \$35 by itself • Accessories: ○ 2A Micro USB power supply costs around \$7, and is needed to power the Pi ○ A Micro SD card, costing around \$8, is needed for memory ○ The cost of other peripherals, such as a monitor, mouse, or keyboard, are not factored in because it is assumed someone buying the Raspberry Pi for the computer security purpose already has these items. This brings the total cost of the system to around \$50.
Form Factor	Provides a small, portable platform for development purposes that can be stored and/or set up in a variety of spaces due to its small footprint.
Performance Upgrades	Offers substantial performance upgrades over its predecessors.
Linux Operating System Available	Capable of running small versions of Linux as well as running most of the programs needed for development.
Debian (Raspbian) System Comes Preloaded	Offers a Debian Linux operating system (Raspbian), which provides several advantages over development with a Windows operating system.
Variety of Choices for Intrusion Detection System and Honeypot Software	There is a variety of free honeypot and intrusion detection system software available if utilizing a Linux operating system.
Licensing a Non-Issue	Licensing issues not encountered, as many are not required.
Availability of Tutorials	Raspbian is very similar to Ubuntu, therefore any tutorial for Ubuntu was generally applicable to Raspbian.

Table 5: Advantages of Raspberry Pi 2

Disadvantages

As hindsight can sometimes be painful, *Table 6: Disadvantages of Raspberry Pi 2* lists the disadvantages of working with the Raspberry Pi 2 module, as discovered by the team throughout the course of the spring semester.

Disadvantages of Choosing the Raspberry Pi 2 Module	
Processing Speed & Available RAM	Despite the performance improvements of the Raspberry Pi 2 over its predecessors, the available RAM and processor speed was a major bottleneck when implementing it as an active honeypot with an intrusion detection system. In specific, when executing all third-party processes, if a simple Nmap port scan was conducted on the device it would become severely bogged down: the processor would run at over 90% capacity and RAM would be maxed. This actually makes the device particularly vulnerable to DoS (Denial of Service) and DDoS (Distributed Denial of Service) attacks where the intent is to overload a targeted system.
Software Compatibility with ARM Processors	Not all programs are compatible with an ARM processor. This quickly became apparent to the team with repeated failures to install Snorby, a Snort graphical user interface.
Development Environments not Available	Code development and testing on the board was severely hindered by the non-existence of any type of Ruby on Rails development environment compatible with an ARM processor. This resulted in a majority of the interface code being written in a text editor and tested via a web browser (limited debugging ability, if any).

Table 6: Disadvantages of Raspberry Pi 2

Third Party Software Installation & Interfaces

Third Party Software Installed

Table 7: 3rd Party Software Install List contains information regarding all of the third party software tools installed on the Raspberry Pi 2 module, including each tool's corresponding software dependencies.

Software Tool	Software Version	Description	Software Dependencies	Install Location
Raspbian	2014-09	Linux Operating System	• g++ • libtool • flex • bison • gcc • libnet1 • libnet1-dev • libcrypt-ssleay-perl • libpcre3 • libpcre3-dev • libphp-adodb • libssl-dev • libwww-perl • ntp • php5-cli • php-pear	--
Kippo	0.8	Honeypot	• python-twisted • python-mysqldb Apache2 must be installed first	/usr/src
Kippo-Graphs	0.8	Honeypot Web Visualization	• libapache2-mod-php5 • php5-gd • php5-mysql	/var/www
Snort	2.9.7	Intrusion Detection System	• libpcap-1.6.2 • libdnet-1.12 • daq-2.0.4	/usr/src
Barnyard2	2-1.13	Database Integration for Snort	Snort must be installed first	/usr/src
PulledPork	0.7.0	Security Definition Integration for Snort	Snort & Barnyard2 must be installed first	/usr/src
Apache2	2.2.22	Webserver for Kippo & Kippo-Graphs	• apache2-utils • apache2.2-bin • apache2.2-common • libapache2-mod-php5	/usr/sbin

Software Tool	Software Version	Description	Software Dependencies	Install Location
MySQL	5.5.41-0+wheezy-log	Database	- mysql-client - mysql-server - mysql-common - libmysqlclient-dev - php5-mysql	/usr/bin
Ruby Version Manager (RVM)	1.26.10	Allows Multiple Versions of Ruby to be Installed	none	/home/pi
Ruby	2.2.0	Development Language	Ruby Version Manager installed	/home/pi/.rvm/rubies
Ruby on Rails	4.2.0	Development Architecture	Ruby Installed	/home/pi/gems
Ruby Gems	2.4.5	Ruby Software Library Tool	Ruby Installed	/home/pi/.rvm/rubies
Aanval	7.6	Intrusion Detection, Correlation & Threat Management Console	Updated versions of Snort, MySQL, Apache and PHP must be installed first	/var/www

Table 7: 3rd Party Software Install List

Configuration Information

The following section details the steps taken to properly configure the third party software components that were installed. To further aid the upcoming Cyber Security Team, copies of the configuration files discussed have been attached in the Appendix material for detailed reference.

Configuring Snort, Barnyard2 & PulledPork

There are three files that require specific configuration information in order to properly “tune” Snort, Barnyard2, and PulledPork. All three files are located under the /etc/snort directory and are titled: “snort.conf”, “barnyard2.conf”, and “pulledpork.conf”.

To correctly configure Snort, *Table 8: Snort Configuration* lists the changes that were implemented in the “snort.conf” file below. Please note, additional changes could be made to this configuration file to more finely tune the types of attacks Snort will investigate; however, these changes should be incorporated into the graphical user interface and identified as different intrusion detection settings for Snort. More detailed information regarding Snort settings can be obtained online from the Snort User Manual, available at <http://manual.snort.org>.

Item No.	Change Applied (Exact Syntax)	Description
1	ipvar HOME_NET 127.0.0.1/22	Sets the home address for the Snort sensor as local host, port #22
2	ipvar EXTERNAL_NET !\$HOME_NET	Allows Snort to differentiate between the external network and the internal network (any web address that is not the local host address)
3	portvar SSH_PORTS 22	Indicates which ports to monitor for possible SSH (brute-force hacking) attempts

Item No.	Change Applied (Exact Syntax)	Description
4	var RULE_PATH /etc/snort/rules	Tells Snort where to look for security definitions & rules for packet sniffing
5	var WHITE_LIST_PATH /etc/snort/rules	Tells Snort where to look for rules regarding non-malicious IP addresses (addresses that can be ignored)
6	var BLACK_LIST_PATH /etc/snort/rules	Tells Snort where to look for rules regarding malicious IP addresses
7	config logdir: /var/log/snort	Tells Snort where to write log and binary file output to
8	output unified2: filename snort.log, limit 128	Configures the Snort log output format
9	include \$RULE_PATH/snort.rules	Instructs Snort to utilize the rule file at this location (**note: all other include paths should be commented out, by utilizing a '#' at the beginning of each line)

Table 8: Snort Configuration

Snort no longer supports database integration, and all output (if configured) will be written as a text log file and binary output file, then stored in the directory location specified by the “config logdir” variable contained within the “snort.conf” file (see item number 7 above). This is why Barnyard2 is utilized, as this software add-on facilitates database integration. Barnyard2 takes the binary output files written by Snort, adjusts the information contained within it, and places this information in a database table.

Therefore, in order to properly configure Barnyard2, the location of these binary files provided by Snort must be made available. *Table 9: Barnyard2 Configuration* lists the changes that were implemented in the “barnyard2.conf” file below.

Item No.	Change Applied (Exact Syntax)	Description
1	config utc	Allows UTC timestamps to be utilized for all events
2	config reference_file: /etc/snort/reference.config	Sets the path for the Snort security reference file
3	config classification_file: /etc/snort/classification.config	Sets the path for the Snort security classification file
4	config gen_file: /etc/snort/gen-msg.map	Sets the path for the Snort security generator file
5	config sid_file: /etc/snort/sid-msg.map	Sets the path for the Snort sensor file
6	config event_cache_size: 250	Sets the limit for the Snort event cache to a max of 250 events
7	config logdir: /var/log/snort	Tells Barnyard2 where to locate Snort’s output files (**note: this directory must match the log directory identified by Snort!!)
8	config interface: eth0	Tells Barnyard2 what network interface Snort will inspect data packets on. If this interface name changes this setting must change as well. (**note: at this time Snort does not support wireless network interfaces)

Item No.	Change Applied (Exact Syntax)	Description
9	config daemon	Allows Barnyard2 to be ran as a daemon process, with an identifiable process id
10	config quiet	Helps to limit unnecessary output to the database by not reporting white-listed activity
11	config waldo_file: /etc/snort/barnyard2-log.waldo	Specifies the location of the waldo file, which is used to facilitate data transfer to the database
12	input unified2	Specifies the input file type to use
13	output database: log, mysql, user=snort password=honey dbname=snort host=localhost	Specifies the database to access and user/password/table information needed to gain access

Table 9: Barnyard2 Configuration

Table 10: PulledPork Configuration lists the changes that were implemented in the “pulledpork.conf” file below. Please note that the sensor ID modification files are not currently configured for use by Snort. These files allow multiple intrusion detection sensors to be created and implemented inside of a network, and may be of use if additional honeypots are added in the future.

Item No.	Change Applied (Exact Syntax)	Description
1	rule_path= /etc/snort/rules/snort.rules	Defines the Snort rule path to which updated security definitions should be loaded
2	sid_msg= /etc/snort/sid-msg.map	Defines where PulledPork should place the sensor file for use by Snort
3	sid_changelog=/var/log/sid_changes.log	Defines where to place the log that is used to record changes to security definitions
4	sorule_path=/usr/local/lib/snort_dynamicrules/	Tells PulledPork where to locate Snort’s dynamic rules (for updating purposes)
5	config_path=/etc/snort/snort.conf	Tells PulledPork where to locate Snort’s configuration file
6	distro=Debian-6-0	Defines the operating system distribution (Raspbian is a Debian based operating system)
7	black_list=/etc/snort/rules/iplists/default.blacklist	Tells PulledPork where to load a publicly obtained blacklist of malicious IP addresses
8	IPRVersion=/etc/snort/rules/iplists	Defines the location for the IP Reputation control socket (allows new IP lists to be loaded while Snort is running)
9	enablesid=/etc/snort/enablesid.conf	Points to the sensor ID modification file (**note: not currently being used)
10	dropsid=/etc/snort/dropsid.conf	Points to additional sensor ID information (**note: not currently being used)
11	disablesid=/etc/snort/disablesid.conf	Points to additional sensor ID information (**note: not currently being used)
12	modifysid=/etc/snort/modifysid.conf	Points to additional sensor ID information (**note: note currently being used)

Table 10: PulledPork Configuration

Configuring Apache2, Kippo & Kippo-Graphs

The Apache2 webserver requires minimal configuration changes to be made. *Table 11: Apache2 Configuration* lists the configuration file location as well as the associated change that was made to the Apache2 webserver.

Item No.	Configuration File	Change Applied (Exact Syntax)	Description
1	/etc/apache2/apache2.conf	ServerName localhost	Allows Apache2 to identify the correct IP Address (here the associated name) for the webserver
2	/etc/apache2/conf-available/servername.conf	ServerName localhost	Associated with the variable ServerName in the configuration file, required to host the webserver on localhost address
3	/etc/php5/apache2/php.ini	error_reporting = E_ALL & ~E_NOTICE	Changes error reporting settings for the webserver

Table 11: Apache2 Configuration

The Kippo honeypot software was configured by making changes to the “kippo.cfg” file located in the /usr/src/kippo-0.8 directory. These changes are reflected below by *Table 12: Kippo Honeypot Configuration*.

The honeypot’s virtual file structure (the files made available for manipulation by hackers) was created by utilizing the python pickle format and the createfs.py utility, which is executed by the file titled “fs.pickle”. When this file is utilized, an image is generated of the existing file structure onto which the honeypot has been installed. This file structure is then made available in the Kippo directory, under the folder titled, “honeyfs”.

In addition, added functionality can be implemented by making changings to the file structure under the folder titled “txtcmds” (also available in the Kippo directory). Here, users can create specialized responses that will be displayed to a hacker once a specific command is issued. For instance, a text file labeled “ls” can be placed under the txtcmd/usr/bin directory, when a hacker is present inside the associated “/usr/bin” directory of the honeypot, entering a command that contains “ls” will yield the contents that were placed inside of the associated “ls” text file. Overall, this allows users to implement more honeypot interaction that is made available to hackers during brute force attacks.

Item No.	Change Applied (Exact Syntax)	Description
1	ssh_port = 2222	Defines the port on which Kippo should listen for incoming SSH (brute-force) attacks
2	hostname = nas3	Defines the name of the honeypot, as seen by the attacker

Item No.	Change Applied (Exact Syntax)	Description
3	log_path = log	Defines where in the directory to save log files (in this instance log files are saved to the subdirectory folder titled “log”)
4	download_path = dl	Defines where in the directory to save downloaded (malicious) files
5	contents_path = honeyfs	Defines the directory in which to store the virtual honeypot file system
6	filesystem_file = fs.pickle	Determines which file structure to utilized when creating the virtual honeypot
7	data_path = data	Defines the directory for storing miscellaneous data files (for manipulation by hackers)
8	txtcmds_path = txtcmds	Defines the directory in which to store the text commands (input displayed to hackers when a certain command is entered)
9	sensor_name = kippo_honeypot	Assigns a name to the Kippo Honeypot sensor
10	host = localhost	Defines where the MySQL server is located
11	database = kippo	Defines the MySQL database to write to
12	username = kippo	Defines the MySQL username for accessing the database
13	password = Kippo-password	Defines the MySQL password for accessing the database
14	logfile = kippo-textlog.log	Sets the name for the Kippo database log file

Table 12: Kippo Honeypot Configuration

Kippo-Graph was configured by making changes to the “config.php” file located under the directory /var/www/kippo-graph/. These changes are reflected below in *Table 13: Kippo-Graph Configuration*.

Item No.	Change Applied (Exact Syntax)	Description
1	define(‘DIR_ROOT’, ‘/var/www/kippo-graph’);	Creates the variable “DIR_ROOT”, which is set to the install path
2	define(‘DB_HOST’, ‘localhost’);	Defines where the MySQL server is located
3	define(‘DB_USER’, ‘kippo’);	Defines the MySQL username for accessing the database
4	define(‘DB_PASS’, ‘Kippo-password’);	Defines the MySQL password for accessing the database
5	define(‘DB_NAME’, ‘kippo’);	Defines the MySQL database to write to

Item No.	Change Applied (Exact Syntax)	Description
6	define('DB_PORT', '3306');	Defines the port on which MySQL communicates

Table 13: Kippo-Graph Configuration

Configuring Ruby on Rails

Ruby Version Manager (RVM) is located under the directory /home/pi/.rvm. To utilize Ruby and Ruby on Rails associated commands in the terminal, the associated RVM script must be started. In addition, the Ruby on Rails webserver must be started in order to view the Cyber Cerburi's webpages. Commands for working with Ruby and Ruby on Rails are displayed below in *Table 14: Important Ruby & Ruby on Rails Commands*.

Command Syntax	Description
source /home/pi/.rvm/scripts/rvm	Starts the RVM service in order to execute Ruby and Ruby on Rails commands in the terminal.
rails server	Starts the Ruby on Rails webserver, this command must be execute from within the associated Ruby project directory (**note: this will execute the rails server inside of the terminal, so typing "Ctrl+C" will stop the server)
nohup rails server	Starts the Ruby on Rails webserver in the background (will not be executed in the terminal)

Table 14: Important Ruby & Ruby on Rails Commands

Database Implementation

Configuring the MySQL Database

Table 15: MySQL Database Configuration: Usernames & Passwords lists the MySQL parameters that were configured for the Cyber Cerburi system:

Item No.	Parameter	Description
1	Username: root	The root username assigned (as Administrative Access to all database tables)
2	Password: honey	The root password assigned
3	/usr/src/schemas	File directory for storing MySQL database information
4	Snort Database – name: snort	The name of the Snort associated database (where Barnyard2 writes all information from Snort)
5	Snort Database – username: snort	The Snort username assigned
6	Snort Database – password: honey	The Snort password assigned
7	Kippo Database – name: kippo	The name of the Kippo associated database
8	Kippo Database – username: kippo	The Kippo username assigned
9	Kippo Database – password: Kippo-password	The Kippo password assigned
10	Cyber Cerburi Database – name: cerburi	The name of the Cyber Cerburi associated database
11	Cyber Cerburi Database – username: root	The Cyber Cerburi username assigned

Item No.	Parameter	Description
12	Cyber Cerburi Database – password: honey	The Cyber Cerburi password assigned

Table 15: MySQL Database Configuration: Usernames & Passwords

Cerburi Database Schema

The following database tables have been generated for use by the Cyber Cerburi system, and are based upon the models created for use in the Ruby on Rails web application:

- data
- events
- icmp_hdrs
- kippo_auths
- kippo_clients
- kippo_downloads
- kippo_inputs
- kippo_sensors
- kippo_sessions
- kippo_ttylogs
- reference_systems
- references
- sensors
- sig_classes
- sig_references
- signatures
- tcphdrs
- traces
- udphdrs
- users

The schema for each database table utilized by the Cyber Cerburi interface is further defined in the next sequence of tables. Please note, each database table is associated with a Ruby on Rails model, and each field of the database is associated with a particular hash value (array field) of said model.

Database tables not yet utilized by the Cyber Cerburi interface have not been described.

The 'events' Table Schema

The 'events' table is generated by the Event model in Ruby on Rails, and is filled with information transcribed from the corresponding 'event' table in the existing Snort database.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
sid	BigInt(20)	Sensor ID
cid	BigInt(20)	Threat Class ID
signature	BigInt(20)	Signature ID
time_stamp	DateTime	Timestamp when event occurred

Table 16: Schema for the 'events' Table, for the Associated 'Event' Model

The 'kippo_auths' Table Schema

The 'kippo_auths' table is generated by the KippoAuth model in Ruby on Rails, and is filled with information transcribed from the corresponding 'auth' table in the existing Kippo database.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
session	VarChar(255)	Session ID – generated by Kippo
success	BigInt(20)	Success Value of the hacking attempt (1= success, 0 = failure)
username	VarChar(255)	Username attempted by the hacker
password	VarChar(255)	Password attempted by the hacker
time_stamp	DateTime	Timestamp when event occurred

Table 17: Schema for the 'kippo_auths' Table, for the Associated 'KippoAuth' Model

The 'kippo_inputs' Table Schema

The 'kippo_inputs' table is generated by the KippoInput model in Ruby on Rails, and is filled with information transcribed from the corresponding 'input' table in the existing Kippo database.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
session	VarChar(255)	Session ID – generated by Kippo
time_stamp	DateTime	Timestamp when event occurred
realm	VarChar(255)	Kippo identified hacking classification: password, username, etc.
success	BigInt(20)	Success Value of the hacking attempt (1= success, 0 = failure)
input	VarChar(255)	Command input entered by the hacker

Table 18: Schema for the 'kippo_inputs' Table, for the Associated 'KippoInput' Model

The 'kippo_sessions' Table Schema

The 'kippo_sessions' table is generated by the KippoSession model in Ruby on Rails, and is filled with information transcribed from the corresponding 'session' table in the existing Kippo database.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
id_val	VarChar(255)	Kippo-associated ID: is the Session ID
starttime	DateTime	Timestamp associated with the start of a brute-force attack
endtime	DateTime	Timestamp associated with the end of a brute-force attack
sensor	BigInt(20)	Sensor ID
ip_val	VarChar(255)	The hacker's IP address, as provided by the WhoIS database

Table 19: Schema for the 'kippo_sessions' Table, for the Associated KippoSession Model

The 'sig_classes' Table Schema

The 'sig_classes' table is generated by the SigClass model in Ruby on Rails, and is filled with information transcribed from the corresponding 'sig_class' table in the existing Snort database.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
sig_class_id	BigInt(20)	The Class ID for a detected threat
sig_class_name	VarChar(255)	The name associated with the class (gives a basic description of the detected threat)

Table 20: Schema for the 'sig_classes' Table, for the Associated SigClass Model

The 'signatures' Table Schema

The 'signatures' table is generated by the Signature model in Ruby on Rails, and is filled with information transcribed from the corresponding 'signature' table in the existing Snort database.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
sig_name	VarChar(255)	The name associated with the class (gives a basic description of the detected threat)
sig_class_id	BigInt(20)	The Class ID for a detected threat
sig_priority	BigInt(20)	The priority level of the detected threat
sig_rev	BigInt(20)	The threat's associated reference ID
sig_sid	BigInt(20)	The Sensor ID
sig_gid	BigInt(20)	The Generator ID for the threat definition

Table 21: Schema for the 'signatures' Table, for the Associated Signature Model

The ‘tcphdrs’ Table Schema

The ‘tcphdrs’ table is generated by the Tcphdr model in Ruby on Rails, and is filled with information transcribed from the corresponding ‘tcphdr’ table in the existing Snort database.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
sid	BigInt(20)	Sensor ID
cid	BigInt(20)	Threat Class ID
tcp_sport	BigInt(20)	Source port field in the data-packet
tcp_dport	BigInt(20)	Destination port field in the data-packet
tcp_seq	BigInt(20)	Associated TCP sequence number of the data-packet
tcp_ack	BigInt(20)	Associated TCP acknowledgement number of the data-packet
tcp_flags	BigInt(20)	Associated TCP flags set in the header of the data-packet
tcp_win	BigInt(20)	Associated TCP window size of the data-packet
tcp_csum	BigInt(20)	Associated TCP checksum value of the data-packet

Table 22: Schema for the ‘tcphdrs’ Table, for the Associated Tcphdr Model

The ‘users’ Table Schema

The ‘users’ table is generated by the User model in Ruby on Rails, and is filled with information collected by the Cyber Cerburi’s set-up wizard.

Field	Type	Description
id	Int(11)	Primary Key: auto-incremented row number
name	VarChar(255)	Cyber Cerburi username
password_digest	VarChar(255)	Encryped Cyber Cerburi password

Table 23: Schema for the ‘users’ Table, for the Associated User Model

Ruby on Rails Interface: The Setup-Up Wizard and Dashboard

Background Information: Electing to Work with Ruby on Rails

The Cyber Security Team’s ideology for the graphical user interface was to provide as much autonomy as possible for the Cyber Cerburi system. In addition, a user had to be provided with an easy way to access the system from other computers on his/or her network. With these requirements in mind, the team was able to narrow down a list of possible frameworks on which to build the graphical user interface. After some deliberation, the Cyber Security Team decided that Ruby on Rails would be the best choice for the development of the graphical user interface, as it appeared to be fairly easy to learn and provided a robust library of software tools.

Ruby on Rails (deemed “Rails” for short) is an extension of the Ruby language that provides a representational state transfer (REST) system architecture on which to design web-based applications. REST-based applications are stateless in form, allowing them to be more lightweight (require less bandwidth). In addition, Rails follows a MVC (Model View Controller) structure in its applications (see below in Figure 1: Ruby on Rails Structure), which allows connections to multiple types of databases (MySQL, SQLite, etc.).

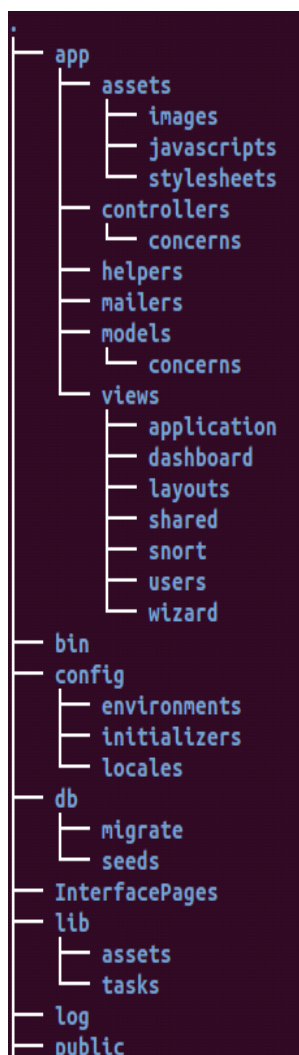


Figure 1: Ruby on Rails Structure

The database selection was based upon the knowledge that the system would be compatible with a variety of database types; however, MySQL was designated as the primary database type due to both its popularity in Linux-based systems, as well as the amount of thorough documentation readily available online.

Rails provides all of the necessary components for creating and running an active web-server project, including database migrations, Ruby Gem installations, and a server for hosting the application. It also provides many tools for testing and deployment of applications; however, the team was not able to utilize many of these testing tools, and they are not included in the scope of this document.

Utilizing the REST architecture, Rails implements a very specific structural layout, which is displayed below in Figure 1. This architecture allows Rails to implement a dynamic way of building applications (albeit slightly confusing at first) but overall provides a very clean graphical user interface.

The Cyber Cerburi system consists of two separate software pieces: a setup wizard and a dashboard. The set-up wizard gathers all the necessary information from the user for the application, while the dashboard provides the user with an encompassing view of the overall activity detected on the network by the honeypot.

The Set-Up Wizard

The thought process for the design of Cyber Cerburi's setup wizard was as follows: when a user plugs in the system and powers it on for the first time, it will gather the necessary information and automatically set all of the required configuration parameters. To correctly configure the system, only a few pieces of information are required from the user. These key pieces are: a username/password for system access and an "oinkcode" for updating Snort definitions.

The setup wizard consists of two main pages. The first page (titled as "index") asks the user for a username and password for accessing the application, and the second page (titled as "oinkcode") asks for the user's "oinkcode" after providing instructions on how to acquire one.

Once the required information is gathered from the user it is saved to the MySQL database and a flag is set (represented by a file in the root directory) that makes the setup page no longer the default homepage for the application. This means that the setup wizard will be displayed as the homepage until a username, password, and "oinkcode" are supplied by the user. After this information is acquired and the set-up process has been completed, this page will no longer be displayed. Figure 2: Set-up Wizard Page displays a screenshot of the setup wizard below.

Welcome to CyberCerburi

[Home](#) [Help](#) [About](#)

Welcome to your new Honeypot

Over the next few screens we will gather some information from you to setup your new honeypot.
Let's start off with the admin username and password

Name

Password

Password confirmation

[Create Account](#)

Cyber Cerburi copyright 2015 CSU Cyber Security Team [About](#)

Figure 2: Set-up Wizard Page

Rails applications work by placing the basic HTML style inside of the 'application' view Ruby file (located at `/app/layout/application.html.erb`), and each page that is accessed by a user is rendered from inside of this file. The code for the application layout is displayed below in Figure 3: Application Layout,

Referencing a Style Sheet. This code sets up the basic background, header, and footer for the page, after which it then calls upon the current page to be rendered (see line number 19, where 'yield' is called). This functionality allows developers to easily change the style of a website without having to change every HTML file.

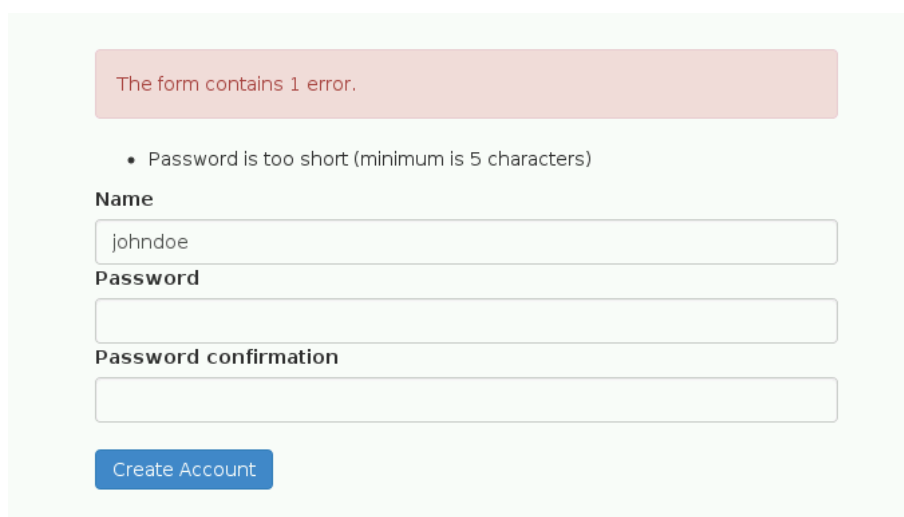
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title><%=full_title(yield(:title))%></title>
5     <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track' => true %>
6     <%= javascript_include_tag 'application', 'data-turbolinks-track' => true %>
7     <%= javascript_include_tag "application" %>
8     <%= javascript_include_tag "//www.google.com/jsapi", "chartkick" %>
9     <%= csrf_meta_tags %>
10  </head>
11  <body>
12    <%= render 'layouts/header' %>
13    <div class="container">
14      <div class="row col-md-12">
15        <% flash.each do |message_type, message| %>
16          <div class="alert alert-<%= message_type %>"><%=message%></div>
17        <% end %>
18      </div>
19      <%= yield %>
20      <%= render 'layouts/footer' %>
21    </body>
22  </html>
```

Figure 3: Application Layout, Referencing a Style Sheet

Username & Password (“Index”) Form

The index form utilizes the Ruby gem “Bcrypt” to encrypt passwords. The password fields located inside the form are setup so that they do not actively display what the user is typing and the data that is transmitted is encrypted. This is just one step the team has taken to ensure user data is properly protected.

After the user enters and confirms a password, these two fields are then compared to ensure that they are the same. The Bcrypt gem also validates that the password meets a certain level of complexity (see Figure 5: Validating Password Requirements for the code implemented in the /app/models/user.rb file). Below, Figure 4: User Password Validation displays an example of a failed validation attempt by a user.



The form contains 1 error.

- Password is too short (minimum is 5 characters)

Name

Password

Password confirmation

Create Account

Figure 4: User Password Validation

Once a valid password is accepted, it is entered into the database as a secure hash of the original password. The code for the index page is displayed below in Figure 6: Storing a Username & Password; this code is located in the /app/view/wizard/index.html.erb file.

```
1 class User < ActiveRecord::Base
2   validates :name, presence: true, length: {maximum: 20}
3   has_secure_password
4   validates :password, presence: true, length: {minimum: 5}
5 end
```

Figure 5: Validating Password Requirements

```
11 <div class="row">
12   <div class="col-xs-6 col-xs-offset-3">
13     <%= form_for (@user) do |f| %>
14       <%=render 'shared/error_messages' %>
15
16       <%=f.label :name , class: 'text-left'%>
17       <%=f.text_field :name, class: 'form-control' %>
18
19       <%=f.label :password %>
20       <%=f.password_field :password, class: 'form-control' %>
21
22       <%=f.label :password_confirmation %>
23       <%=f.password_field :password_confirmation, class: 'form-control' %>
24       <br>
25       <%=f.submit "Create Account", class: " btn btn-primary" %>
26     <%end%>
27   </div>
28 </div>
29 </div>
```

Figure 6: Storing a Username & Password

By utilizing this method, the actual password created by the user is never seen by the database, as only a hashed version is stored. Therefore in the event that the Cyber Cerburi’s database is compromised by a hacker, the stored passwords would have to be compared against a list of possible passwords. This process is both computationally expensive and time consuming. The information acquired by the index page (a username and hashed password) is then stored as an entry in the “user” table of the cerburi database.

Oink-Code Form

An oink-code is a generated code required by the PulledPork software add-on to load and update security definitions found in the Snort database. This code is obtained through registering as a user at Snort.org, and does not cost any money. The oink-code form instructs the user (by way of screenshots) on how to properly register at Snort.org, and how to locate the generated code. Once the user provides this code to the Cyber Cerburi system by entering it into a textbox, it is then updated in the PulledPork configuration file (which is located in the /etc/snort/ directory).

The images used to instruct the user are stored within the Rails “assets” directory, in the “image” folder. All assets utilized by the graphical user interface are stored within this directory and can be accessed from any code within the project by simply utilizing a command. Figure 7: Invoking Assets (Images) displays

an example of how to utilize image insertion tags to invoke an asset (in this case a set of screenshots), while Figure 8 through Figure 12 displays the rendered images found on the oink-code form.

```
11 <div class="row col-md-8 col-md-offset-2">
12   <ol>
13     <h2><li>Go to: <a href="http://www.snort.org/users/sign_up" target="_blank">Snort User Signup</a></li></h2>
14     <h2><li>Fill out the form for new user signup</li></h2>
15     <%= image_tag "snort_sign_up.png", alt:"snort sign up page", width:"800", height:"600"%>
16     <h2><li>You will receive an email asking you to confirm your account, click on the link to confirm and go bac
17     <%= image_tag "snort_confirm.png", alt:"Snort email confirmation", width:"800",height:"600"%>
18     <h2><li>Sign in to your new account</li></h2>
19     <%= image_tag "snort_sign_in.png", alt:"Snort sign in page", width:"800",height:"600"%>
20     <h2><li>Click on your email in the upper right corner to go to your account page</li></h2>
21     <%= image_tag "snort_main.png", alt:"Snort main page", width:"800",height:"600"%>
22     <h2><li>Go to Oinkcode tab and copy the provided oinkcode then paste it into the box on this page.</li></h2>
23     <%= image_tag "snort_code.png", alt:"Snort account page", width:"800",height:"600"%>
24   </ol>
25 </div>
```

Figure 7: Invoking Assets (Images)

Oink Code Setup

Now we will get you an oinkcode so that you can keep your honeypot up to date with the latest definitions from [Snort](#).

1. Go to: [Snort User Signup](#)
2. Fill out the form for new user signup

Sign up

Email
Please enter your Email address

Password

Password confirmation

☒ Agree to [Snort license](#)

[Sign in](#)
Didn't receive confirmation instructions?

Figure 8: Instructing Users on How to Obtain an Oink-Code: Steps 1 & 2

3. You will receive an email asking you to confirm your account, click on the link to confirm and go back to the sign in page.

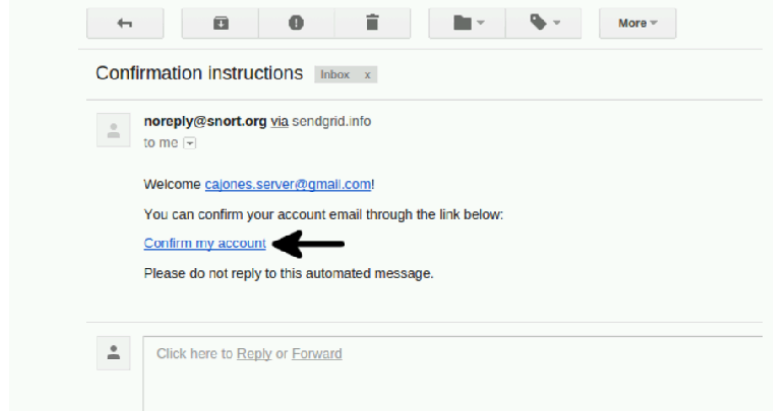


Figure 9: Instructing Users on How to Obtain an Oink-Code: Step 3

4. Sign in to your new account

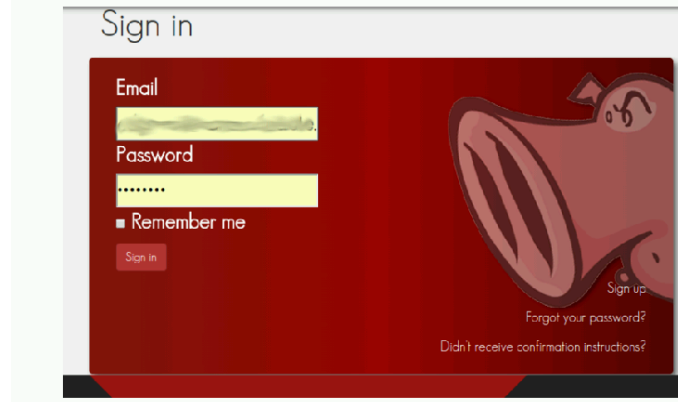


Figure 10: Instructing Users on How to Obtain an Oink-Code: Step 4

5. Click on your email in the upper right corner to go to your account page

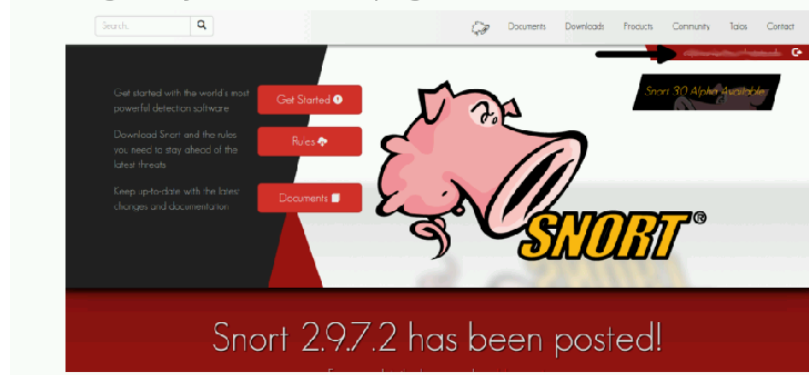


Figure 11: Instructing Users on How to Obtain an Oink-Code: Step 5

6. Go to Oinkcode tab and copy the provided oinkcode then paste it into the box on this page.

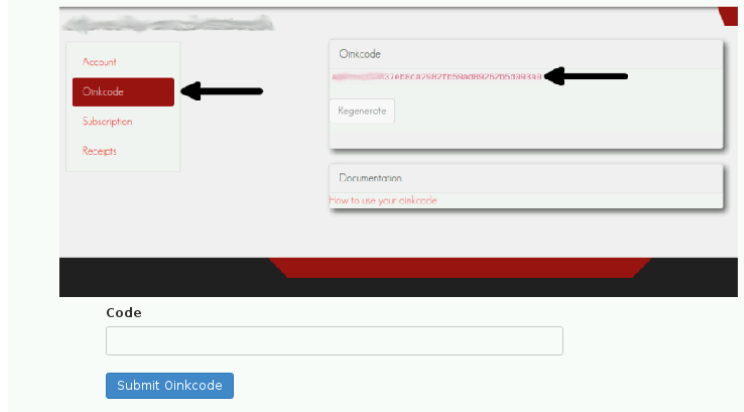


Figure 12: Instructing Users on How to Obtain an Oink-Code: Step 6

Rails offers the ability to embed Ruby code inside of an HTML page in order to perform behind the scenes tasks. In this case, an algorithm was embedded inside of the form in order to search through the PulledPork configuration file and locate the appropriate locations in which to insert the user's generated oink-code. This algorithm is displayed below in Figure 13: Locating Where to Place the Oink-Code.

```
39 def save_code(code)
40   if(code.empty?)
41     return false
42   end
43   snortrules = "snortrules-snapshot.tar.gz"
44   opensource = "opensource.gz"
45   cont = []
46   File.open("/etc/snort/pulledpork.conf", 'r') do |f|
47     f.each_line do |l|
48       if(l.include?(snortrules) || l.include?(opensource))
49         str = l.split(' ')
50         str[2] = code
51         cont.push(str.join(' '))
52       else
53         cont.push(l)
54       end
55     end
56   end
57   File.open("/etc/snort/pulledpork.conf", 'w') do |f|
58     cont.each {|c| f.puts(c)}
59   end
60   true
61 end
```

Figure 13: Locating Where to Place the Oink-Code

Once the oink-code is successfully inserted into the PulledPork configuration file the Cyber Cerburi system then deletes a flag file titled "first.run" located inside the home directory. This file contains no information inside of it; it is merely a way in which the system can determine if it has been configured by a user. For instance, if the first run file exists (the flag has not been removed), the user will be shown the username/password form instead of the dashboard. It is assumed that the user will perform both the username/password generation and supply the necessary oink-code in one session, as additional functionality to determine if user information exists (minus an oink-code) has not been provided. This functionality could be added onto the software in later revisions, however.

If the first run file no longer exists in the home directory, the dashboard form will be loaded. This page routing function is performed within the “routes.rb” file located in the config directory. All page routes allowed by the application are defined within this file.

For instance, if the address “localhost:3000\wizard” is entered into the system’s web browser, the user will be directed to the first page of the setup wizard, or more concisely, to the page that the index method in the wizard’s controller defines (index.html.erb), as the system defines its root form as either “wizard#index” (the setup wizard form) or “dashboard#index” (the dashboard form) based on the presence of the first run file. Figure 14: The Routes File displays the contents of the routes file below.

```
1 Rails.application.routes.draw do
2   get 'snort'      => 'snort#index'
3
4   get 'users/new'  => 'wizard#new'
5   get 'users'      => 'wizard#index'
6   post 'users'     => 'wizard#create'
7   get 'users/create'
8
9   if(File.exists?( './first.run'))
10    root 'wizard#index'
11  else
12    root 'dashboard#index'
13  end
14
15  get 'wizard'      => 'wizard#index'
16  get 'wizard/oink' => 'wizard#oink'
17  get 'oinks/new'   => 'wizard#oink'
18  get 'oinks'       => 'wizard#index'
19  post 'oinks'      => 'wizard#oinkcode'
20  get 'oinks/create'
21  get 'home'        => 'dashboard#index'
22  get 'about'       => 'application#about'
23  get 'help'        => 'application#help'
24
25 end
```

Figure 14: The Routes File

Dashboard Form

The Cyber Cerburi dashboard is the primary interface in which to view information obtained by Kippo and Snort. A screenshot of the dashboard is displayed below in Figure 15: Cyber Cerburi Dashboard. Numerous charts were included on this page in order to provide an encompassing view on the state of the Cyber Cerburi system. Kippo hacking attempts, Snort event totals, as well as basic data packet information (source and destination ports) are all described. The user has also been provided with a “sidebar” of basic information to explain different trends that may be seen in activity.

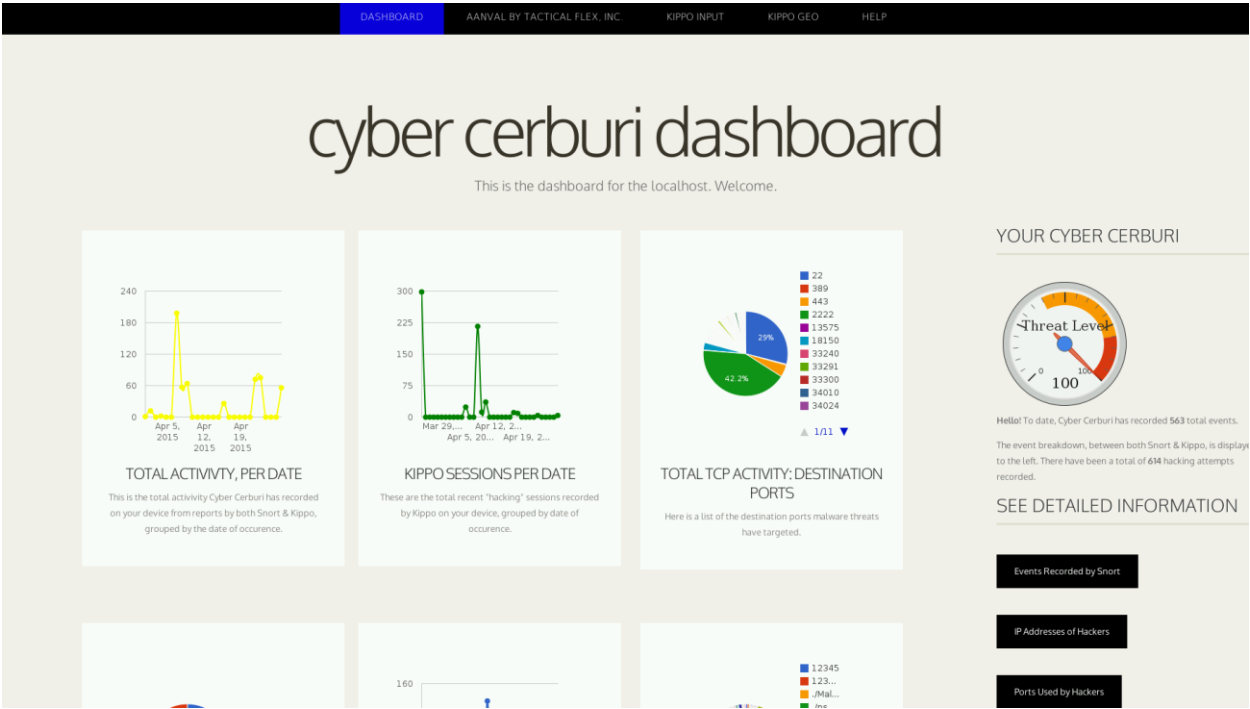


Figure 15: Cyber Cerburi Dashboard Page

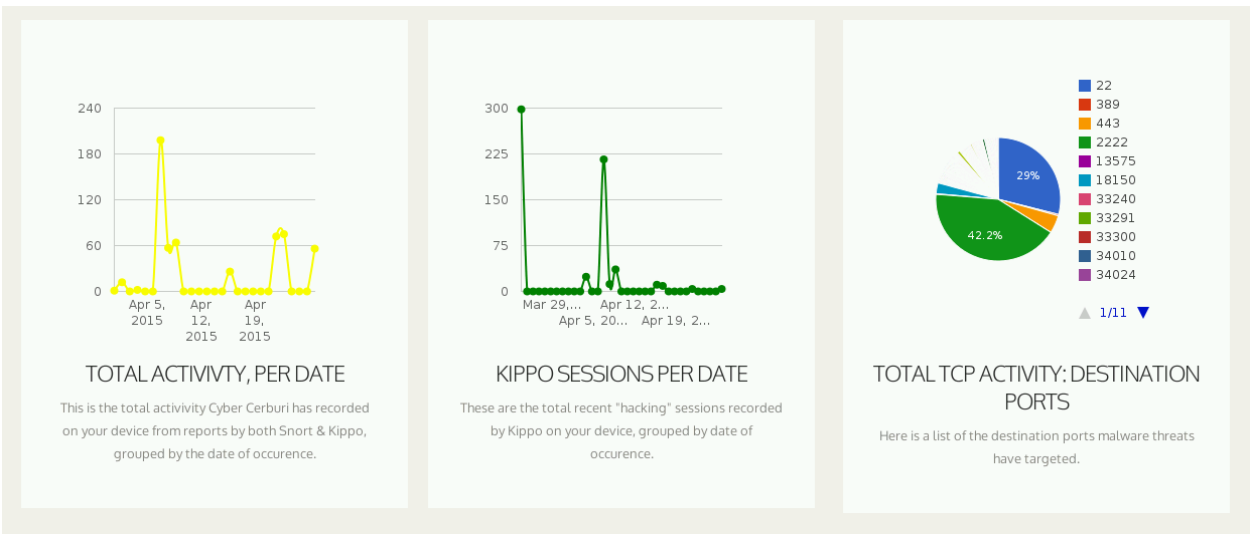


Figure 16: Cyber Cerburi Dashboard, First Row of Charts

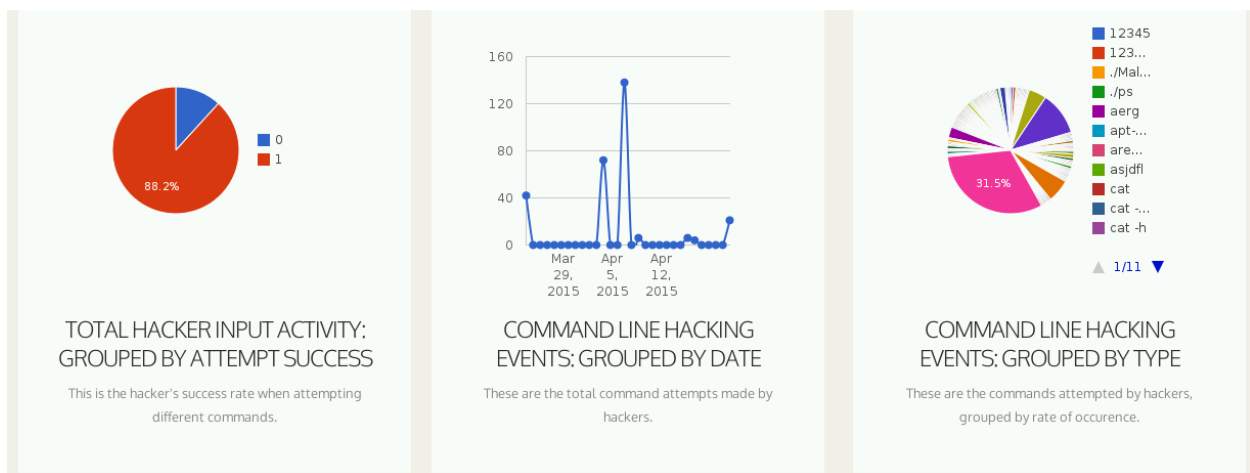


Figure 17: Cyber Cerburi Dashboard, 2nd Row of Charts



Figure 18: Cyber Cerburi Dashboard, 3rd Row of Charts

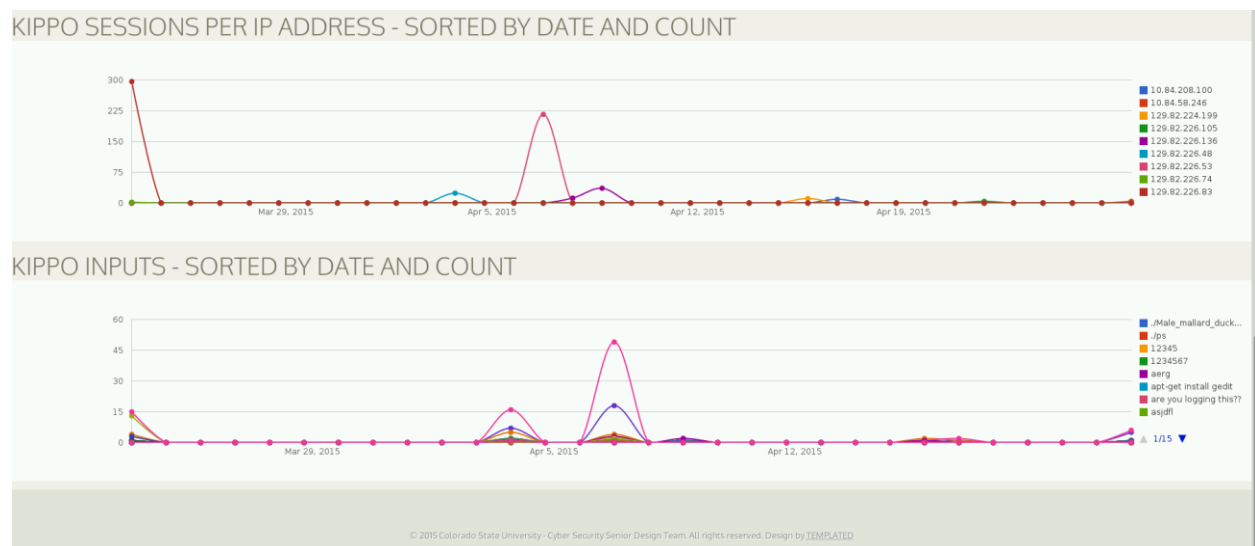


Figure 19: Cyber Cerburi Dashboard Page - Kippo Activity

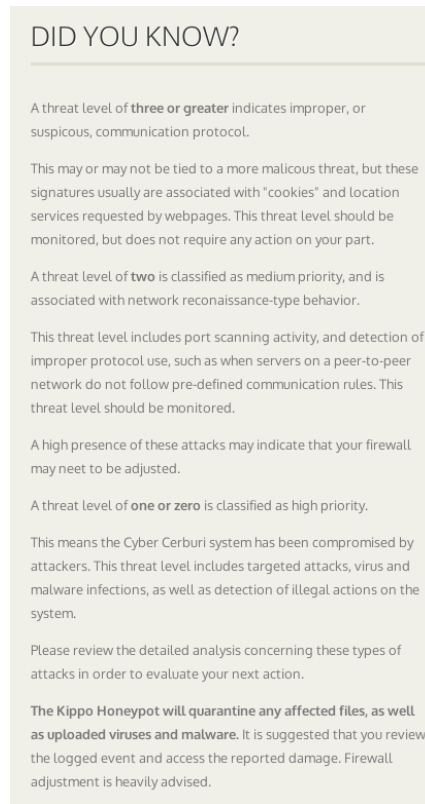


Figure 20: Cyber Cerburi Dashboard, Information Provided in Sidebar

Detail Pages

The following section lists the detailed information pages available through the Cyber Cerburi's dashboard interface. Each detailed page has been equipped with a table showing pertinent information gathered from select database tables, as well as a corresponding "bubble chart", which allows multiple parameters to be grouped together in order to look for possible patterns in activity.

Snort Event Detail

The Snort Event Detail page contains filtered content related to each event recorded by Snort. Code has been added that sorts through numerous Snort database tables in order to group each event's identification and signature number with the corresponding threat identification number, threat name, priority level and time stamp. Threats have been further color coordinated to allow easy viewing for the user. A threat level of one or zero indicates that the Kippo honeypot has been breached, whereas a threat level of two indicates port scanning, or reconnaissance activity, and threat levels of three or greater indicates improper or suspicious communication protocol occurrences.

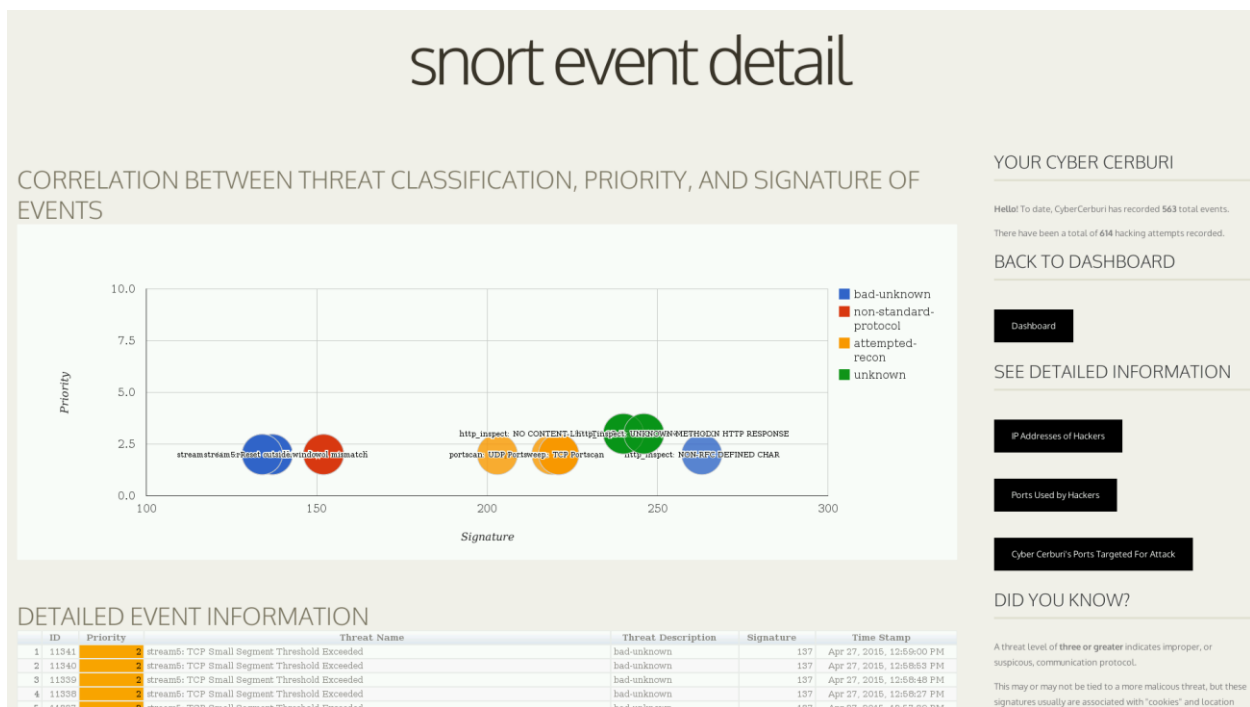


Figure 21: Snort Event Detail Page

72	11268	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 6:10:40 PM
73	11267	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 6:01:39 PM
74	11266	3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	240	Apr 23, 2015, 6:00:07 PM
75	11265	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:55:02 PM
76	11264	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:55:02 PM
77	11263	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:53:20 PM
78	11262	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:46:10 PM
79	11261	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:46:07 PM
80	11260	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:38:29 PM
81	11259	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:37:36 PM
82	11258	3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	240	Apr 23, 2015, 5:35:52 PM
83	11257	3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	240	Apr 23, 2015, 5:35:49 PM
84	11256	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:33:54 PM
85	11255	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:31:39 PM
86	11254	3	http_inspect: UNKNOWN METHOD	unknown	246	Apr 23, 2015, 5:31:35 PM
87	11253	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:30:12 PM
88	11252	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:25:08 PM
89	11251	3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	240	Apr 23, 2015, 5:20:26 PM
90	11250	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:12:15 PM
91	11249	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:11:12 PM
92	11248	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:10:40 PM
93	11247	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:10:40 PM
94	11244	3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	240	Apr 23, 2015, 5:06:46 PM
96	11245	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:06:46 PM
96	11246	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:06:46 PM
97	11243	3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	240	Apr 23, 2015, 5:06:42 PM
98	11242	2	portscan: TCP Portscan	attempted-recon	221	Apr 23, 2015, 5:01:31 PM
99	11241	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 5:00:16 PM
100	11240	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 4:58:27 PM
101	11239	2	ssh: Protocol mismatch	non-standard-protocol	152	Apr 23, 2015, 4:47:42 PM
102	11218	2	stream5: Reset outside window	bad-unknown	134	Apr 23, 2015, 4:45:30 PM
103	11226	2	stream5: Reset outside window	bad-unknown	134	Apr 23, 2015, 4:45:30 PM

Figure 22: Snort Event Detail - Data Table

Kippo Event Detail

The Kippo Event Detail page contains filtered content related to each event recorded by Kippo. Code has been added that sorts through numerous Kippo database tables in order to group each session’s individual identification number with the corresponding IP address of the hacker, the start and end times for the session, as well as the password, username, and command line inputs entered (see *Figure 23: Kippo Detail* *Page:* *Overview*).



Figure 23: Kippo Detail Page: Overview

A bubble chart has been added that correlates success rate and commands with the IP address of the hacker in an effort to recognize possible attack patterns (see *Figure 24: Kippo Bubble Chart*).

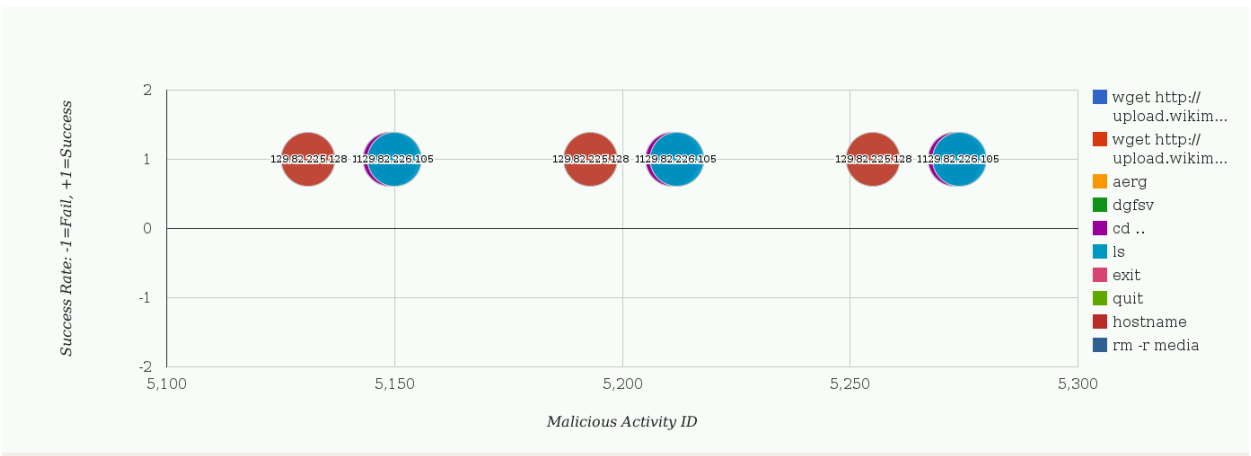


Figure 24: Kippo Bubble Chart

Successful hacking events are also described in detail in the table provided on this page (see Figure 25: Detailed Successful Session Information below).

DETAILED SESSION INFORMATION: SUCCESSFUL EVENTS							
ID	Start Time	End Time	IP Address of Hacker	Session ID	Username	Password	Input
5255	Apr 27, 2015, 7:19:29 PM	Apr 27, 2015, 10:51:25 PM	129.82.225.128	537acb94ed1211e4b0b8b827ebb0bf7	root	123456	wget http://upload.wikimedia.org/wikipedia/commons/2/24/Male_mallard_duck_2.jpg
5255	Apr 27, 2015, 7:19:29 PM	Apr 27, 2015, 10:51:25 PM	129.82.225.128	537acb94ed1211e4b0b8b827ebb0bf7	root	123456	wget https://upload.wikimedia.org/commons/2/24/Male_mallard_duck_2.jpg
5273	Apr 22, 2015, 9:20:21 PM	Apr 22, 2015, 9:34:58 PM	129.82.226.105	61dd181ce93511e49cf7b827ebb0bf7	root	123456	aerg
5273	Apr 22, 2015, 9:20:21 PM	Apr 22, 2015, 9:34:58 PM	129.82.226.105	61dd181ce93511e49cf7b827ebb0bf7	root	123456	dgfsv

Figure 25: Detailed Successful Session Information

Finally, information regarding each field displayed in the table is displayed in the sidebar of the page (see *Figure 26: Informational Sidebar Material*).

DID YOU KNOW?

Many hackers can "mask" their IP addresses by utilizing a virtual private network or a web-proxy service. This should be taken into consideration if you are reporting evidence to officials.

The **Session ID** reported is tied to a specific hacking occurrence recorded by the Kippo Honeypot. All incidents that occurred in a specific session will possess this same identification number.

The **Username** and **Password** reported are those utilized by the hacker to infiltrate your system.

The **Input** reported is the actual hacker activity recorded.

Figure 26: Informational Sidebar Material

Data Packet Detail Pages

Two pages have been added that focus on the data-packet information collected by Snort: the Ports Targeted for Attack and the Malicious Source Ports page. The Ports Targeted for Attack page displays data-packet header information collected per event by Snort (see *Figure 27: Ports Targeted for Attack Detail Page*).

The focus of the bubble chart on this page is to relate the port attacked on the Cyber Cerburi system with the threat classification and the TCP window size, in an effort to possibly visualize event patterns (see *Figure 28: Ports Targeted for Attack Bubble Chart*).

The TCP window size collected in the data-packet's header defines how much data is stored in the buffer before being sent to the application layer. It is common practice to utilize this parameter to carry out DoS (Denial of Service) attacks, or it initiate what is known as "socket stress" on a victim computer by either sending packets with windows sized at zero bytes, or many packets with very small window sizes (around four bytes or less) – as window size is a common tool in handling data flow by the TCP protocol, as well as avoiding network congestion.

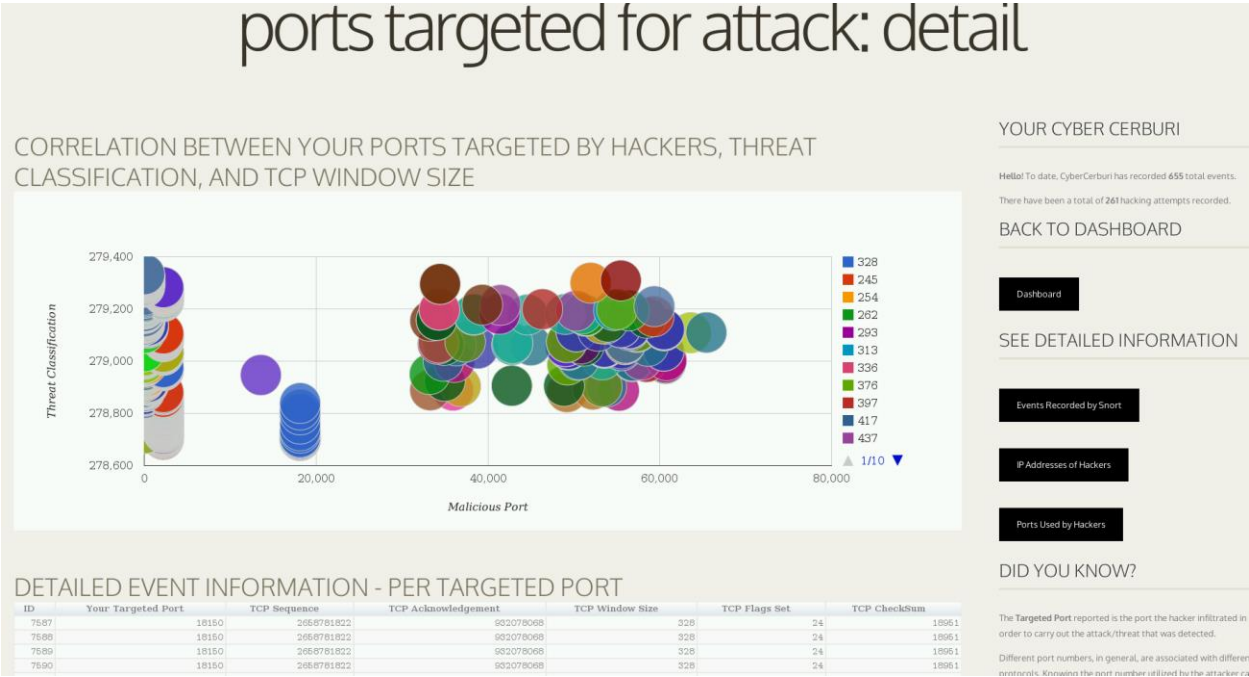


Figure 27: Ports Targeted for Attack Detail Page

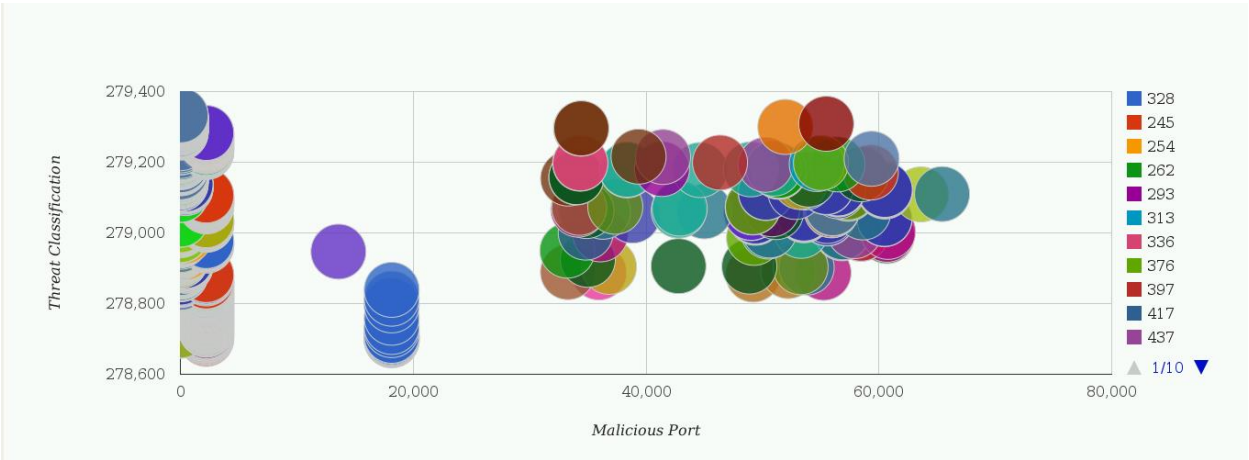


Figure 28: Ports Targeted for Attack Bubble Chart

The data table provided on this page also lists the targeted port on the Cyber Cerburi, with the TCP sequence and acknowledgement numbers contained in the data-packet's header, as well as the window size, the flags (options) set, and the checksum calculation (Figure 29: Ports Targeted for Attack Data Table).

DETAILED EVENT INFORMATION - PER TARGETED PORT

ID	Your Targeted Port	TCP Sequence	TCP Acknowledgement	TCP Window Size	TCP Flags Set	TCP CheckSum
7587	18150	2658781822	932078068	328	24	18961
7588	18150	2658781822	932078068	328	24	18961
7589	18150	2658781822	932078068	328	24	18961
7590	18150	2658781822	932078068	328	24	18961
7591	2222	1706404273	4212596386	245	24	36232
7592	18150	2658781822	932078068	328	24	18961

Figure 29: Ports Targeted for Attack Data Table

The Malicious Source Port Detail page is very similar, however, here the port number utilized by the hacker (or the source address as listed in the data-packet header) is used instead. The bubble chart

included, as well as the data table reflect the same fields, but now correlated with the hacker’s port number.

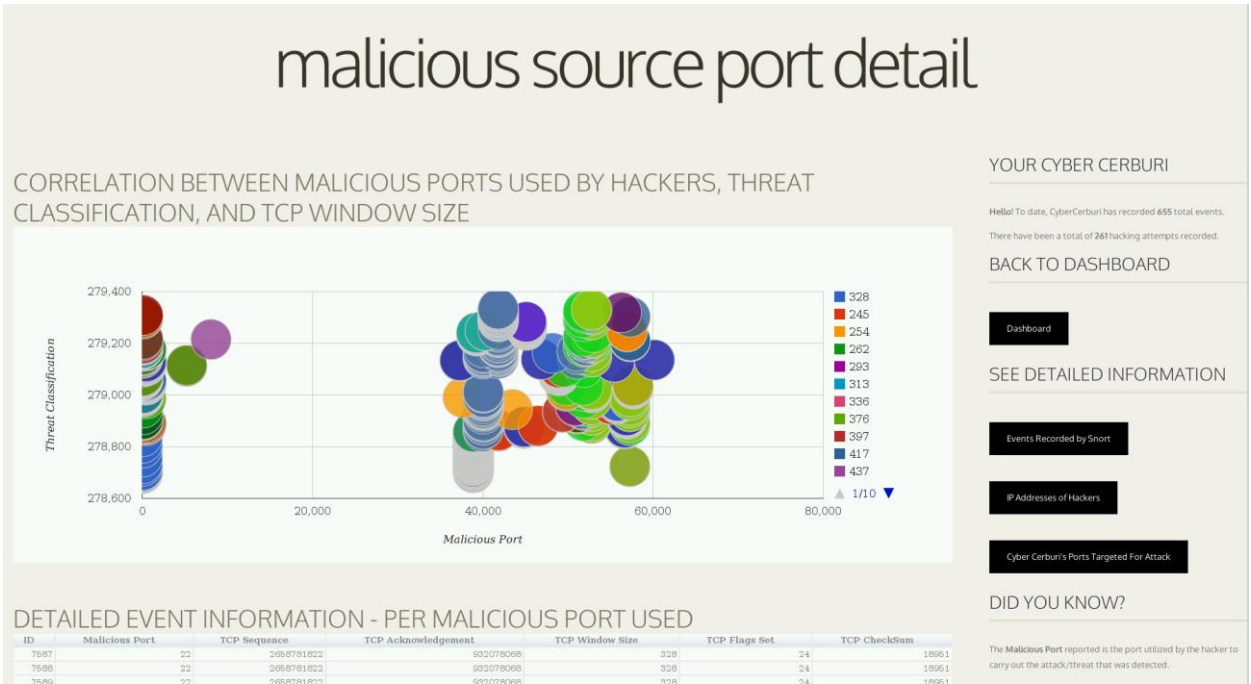


Figure 30: Malicious Source Port Detail

Both pages also provide very similar information material for the user that discusses what each field listed in the data table means, and why it is important (see *Figure 31: Sidebar Material for Both Pages*).

DID YOU KNOW?

The **Malicious Port** reported is the port utilized by the hacker to carry out the attack/threat that was detected.

Different port numbers, in general, are associated with different protocols. Knowing the port number utilized by the attacker can help narrow-down the possibilities when determining the type of threat that was detected -- simply based on protocol association.

The **TCP Sequence** of a threat is specifically tied to the type of threat detected. Some Denial of Service attacks, as well as Trojan malware, are tied to a specific TCP sequence.

Together, the **TCP Acknowledgement**, along with the type of **TCP Flags** set, can be used to "fingerprint" some types of port scans, such as Nmap.

The **TCP Window Size** and the **TCP Checksum** can allude to the amount and "type" of data detected per packet. This information aides in providing a more encompassing threat signature.

The **Targeted Port** reported is the port the hacker infiltrated in order to carry out the attack/threat that was detected.

Different port numbers, in general, are associated with different protocols. Knowing the port number utilized by the attacker can help narrow-down the possibilities when determining the type of threat that was detected -- simply based on protocol association.

The **TCP Sequence** of a threat is specifically tied to the type of threat detected. Some Denial of Service attacks, as well as Trojan malware, are tied to a specific TCP sequence.

Together, the **TCP Acknowledgement**, along with the type of **TCP Flags** set, can be used to "fingerprint" some types of port scans, such as Nmap.

The **TCP Window Size** and the **TCP Checksum** can allude to the amount and "type" of data detected per packet. This information aides in providing a more encompassing threat signature.

Figure 31: Sidebar Material for Both Pages

Help/FAQ Page

A Help / Frequently Asked Questions Page was added to answer common questions a user may have when becoming acquainted with the Cyber Cerburi interface. A table of contents was created as a sidebar to allow the user to quickly peruse and navigate to the material available (see *Figure 32: Cyber Cerburi Help/FAQ Page*).

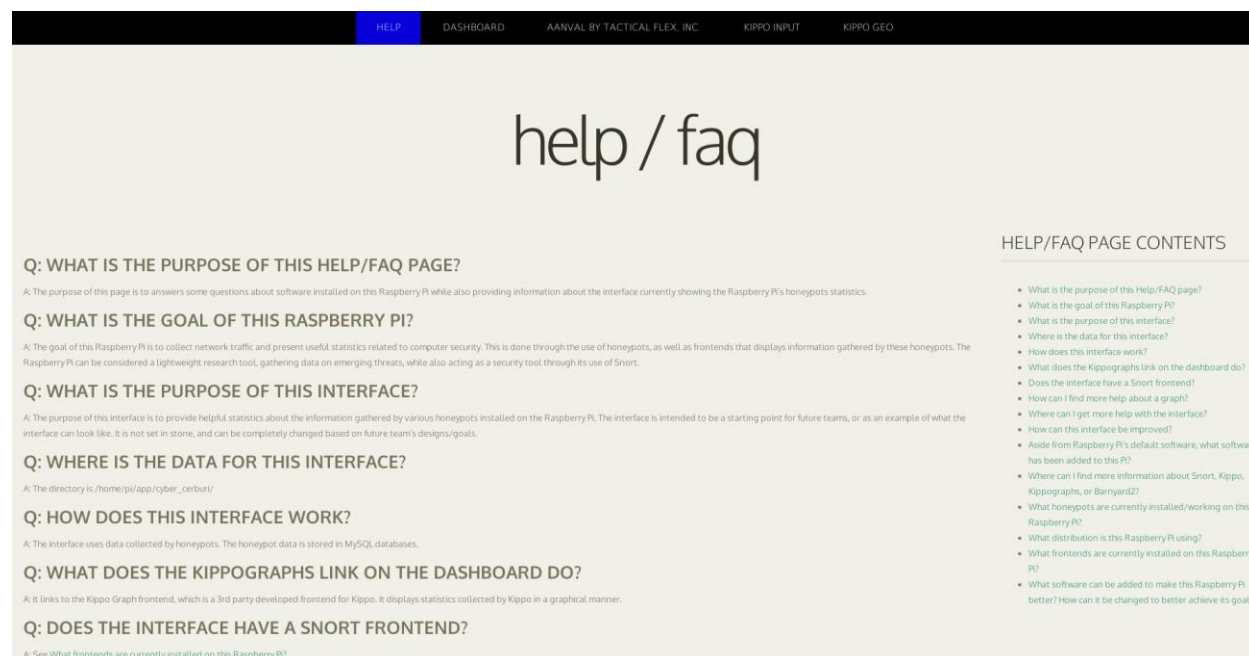


Figure 32: Cyber Cerburi Help/FAQ Page

Kippo-Graph Pages

The database entries that Kippo saves can also be accessed through the Kippo-Graphs web interface installed. Kippo-Graphs is a PHP-based web interface that requires Apache2 (a webserver) to operate. The graphical user interface provided includes a variety of charts, graphs and table visualizations created by the information provided by Kippo. The Cyber Security Team has reformatted the HTML style of Kippo-Graphs to match that of the dashboard, as well as omitted a few non-functioning graphs and charts.

Information displayed is data that has been collected directly from the Kippo database table. Detailed information is displayed regarding hacking activity, such as successful and failed command line inputs, password and username inputs, as well as connections per brute-force attempt (SSH) – to name a few. Kippo-Graphs also utilizes geolocation look-up of the source IP address of the hacker. This information may or may not be the accurate location of the hacker (as a web-proxy service or a Virtual Private Network may have been utilized).

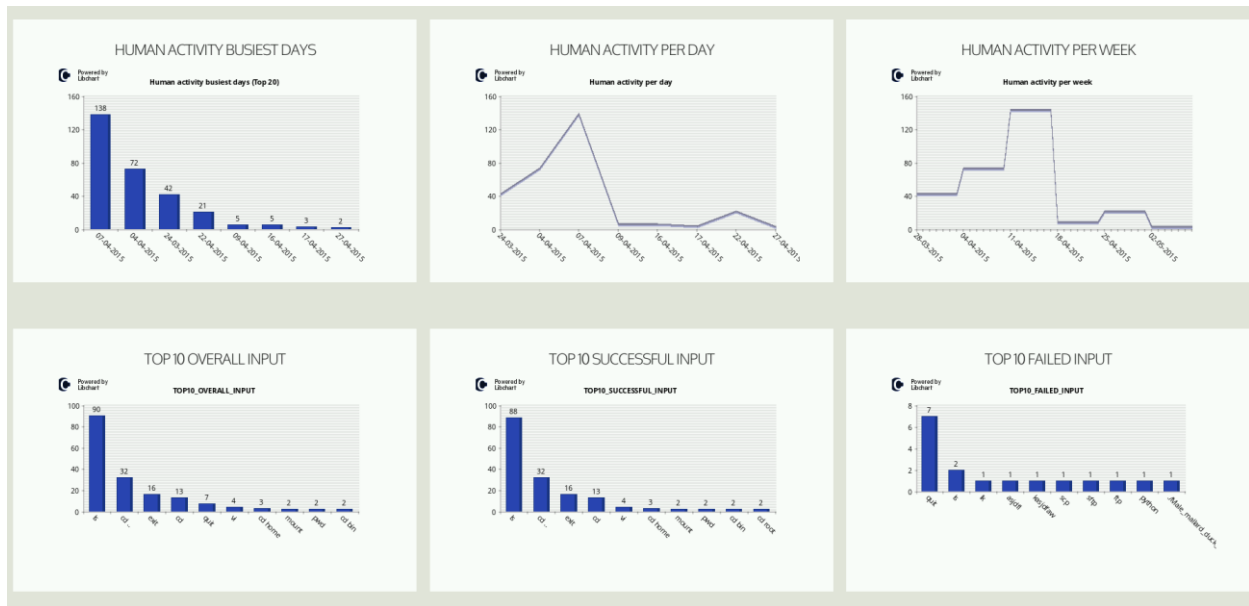


Figure 33: Kippo-Graph Pages

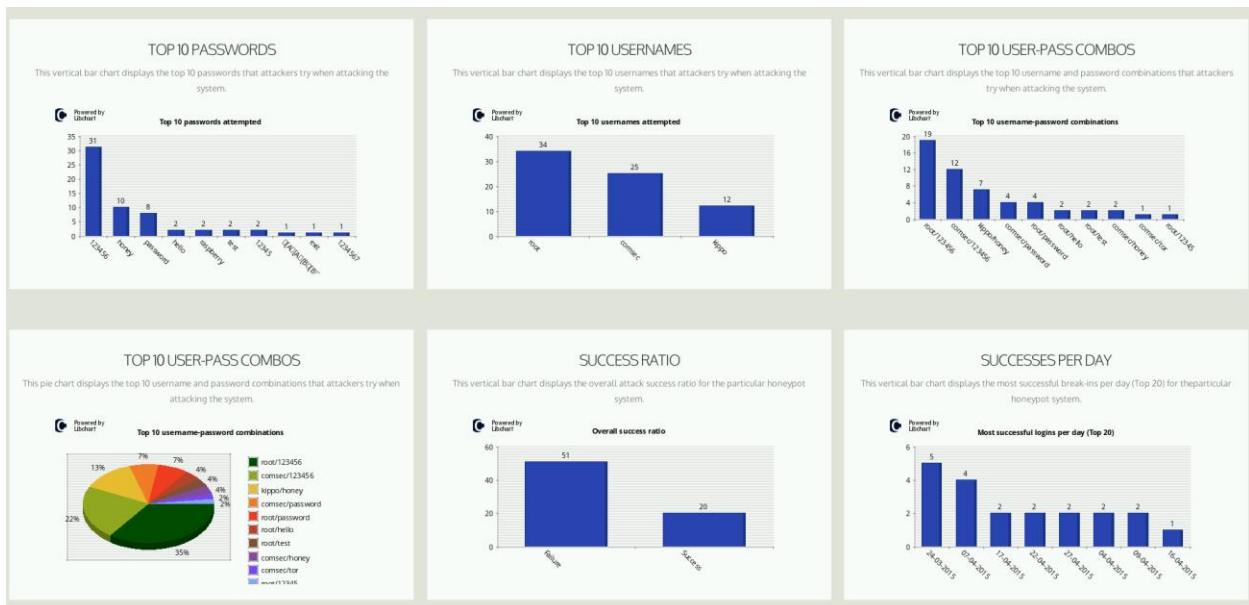


Figure 34: Kippo-Graph Pages



Figure 35: Kippo-Graph Pages

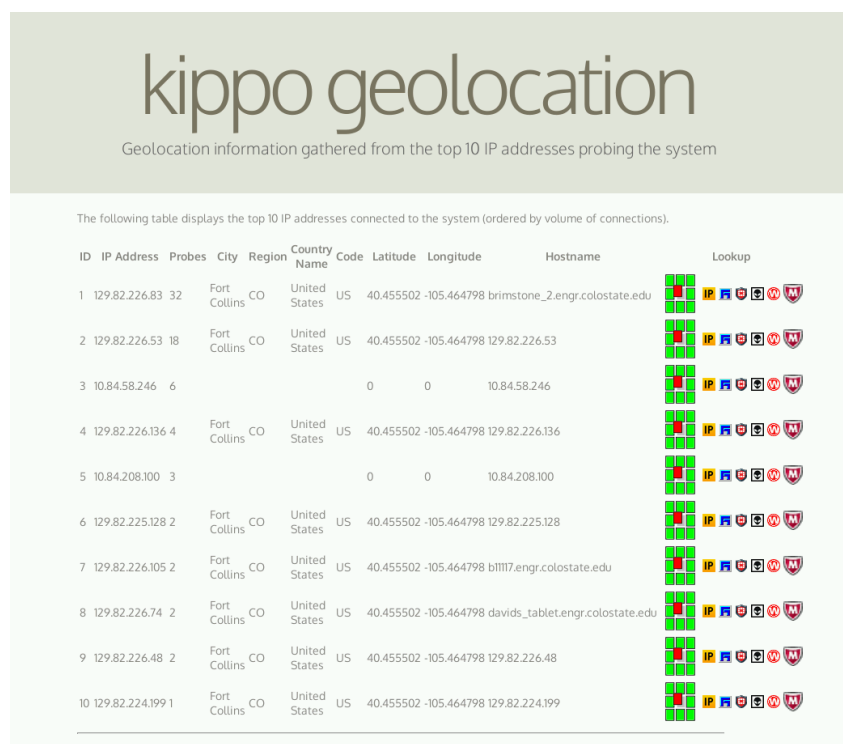


Figure 36: Kippo-Geolocation Pages

Aanval by Tactical Flex, Inc.

The Cyber Security Team has been granted the use of an SAS Enterprise License for the 2015 – 2016 academic year. Aanval is a commercially available, Snort-based, intrusion detection, correlation, and threat management system used by cyber security professionals worldwide.

Configuring Aanval

To utilize Aanval, the processing engines must be started through the terminal shell (or through a bash script). Aanval has been installed in the /var/www/ directory, and can be started by navigating to the subdirectory /apps/ in the terminal and entering the command: “perl ./idsBackground.pl –start”.

A “readme.txt” file has been placed on the desktop of the Cyber Cerburi system that contains the default password and username to access the Aanval web interface.

Aanval Homepage

The Aanval homepage can be found through navigation buttons made available on both the dashboard and detail pages, see *Figure 37: Navigating to Aanval's Homepage* below.

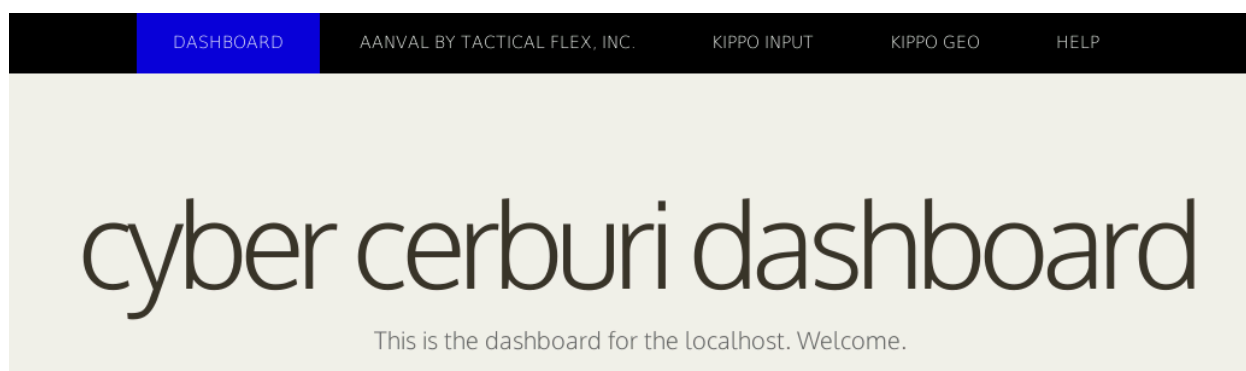


Figure 37: Navigating to Aanval's Homepage

Selecting to navigate to the Aanval homepage will direct the user to log into the system, as seen in *Figure 38: Aanval Log-In Page*.

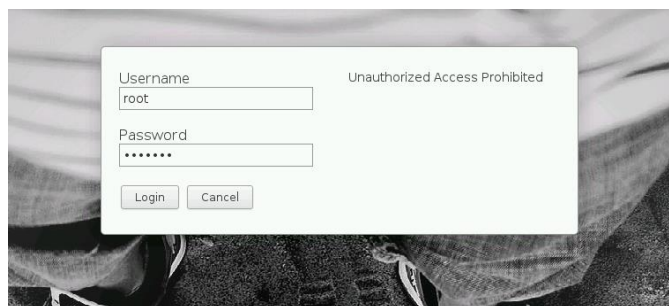


Figure 38: Aanval Log-In Page

After successfully logging into the system, a comprehensive dashboard interface is displayed. The most recent events, as well as a small activity trend are pictured below in *Figure 39: Aanval Homepage*. There are also additional buttons provided in the top left to navigate to more detailed viewing and user settings pages.

Aanval's dashboard allows users to filter events based on the time stamp of occurrence, options to view only the last hour, the last six hours, or the last twelve hours are provided. Each entry also provides the

type of threat recorded, its priority level and description, as well as the sensor impacted, IP addresses and ports involved.

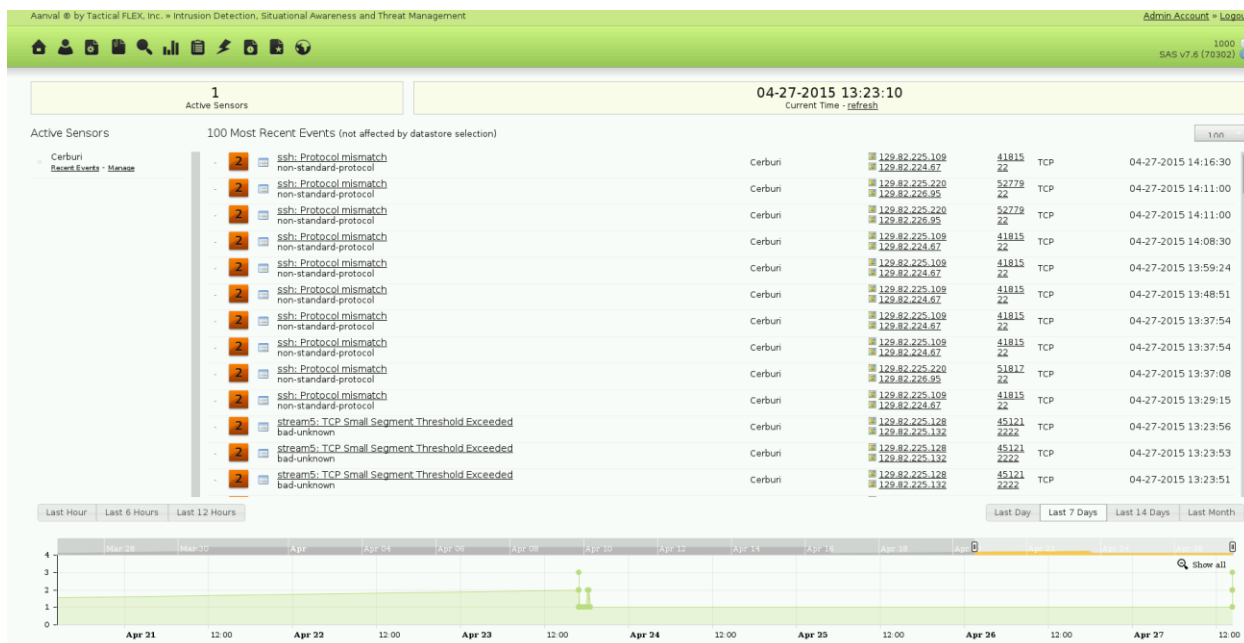


Figure 39: Aanval Homepage

The Event Timeline Browser is another useful tool provided by Aanval, as this view allows you to peruse the different Snort recorded events by date, sorted visually by each threat's risk level. This allows users to quickly visually access both the rate and type of activity that has occurred on the network (see *Figure 40: Aanval's Event Timeline Browser* below).

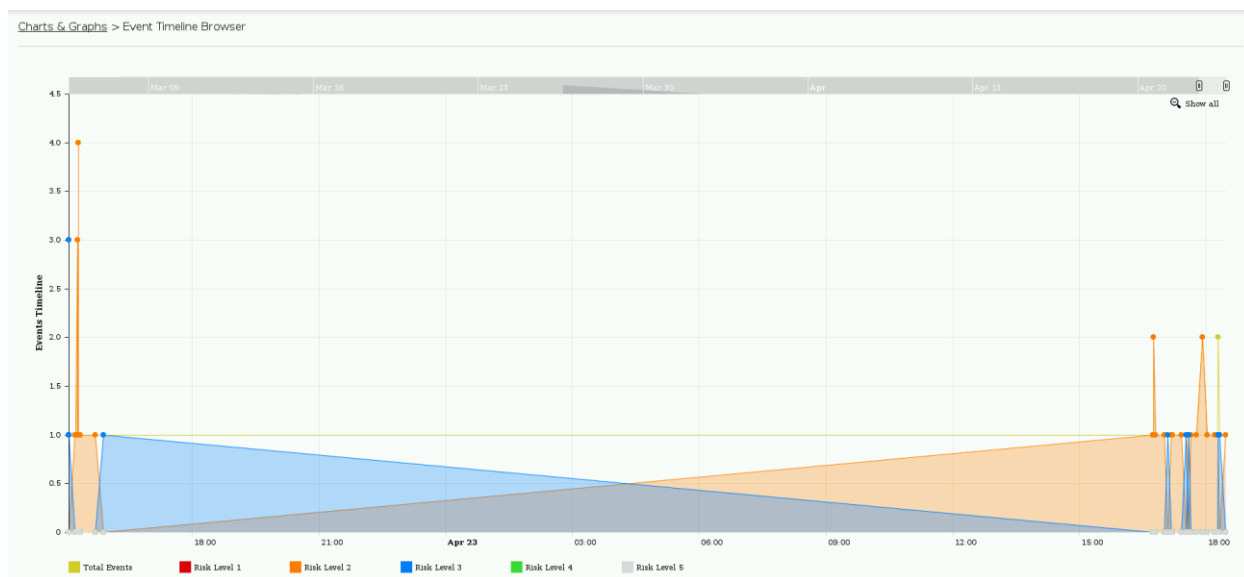


Figure 40: Aanval's Event Timeline Browser

The Frequent Offender page (see *Figure 41: Aanval's Frequent Offender Page*) was also another favorite of the team, as this provides a service that is not currently supported by the Cyber Cerburi dashboard. Here, frequent offending attackers have been geo-located and ranked by rate of occurrence. This page is

also quite visually pleasing and provides users with pertinent information, while only requiring minimal analytical efforts on the part of the user to understand the information that has been collected.

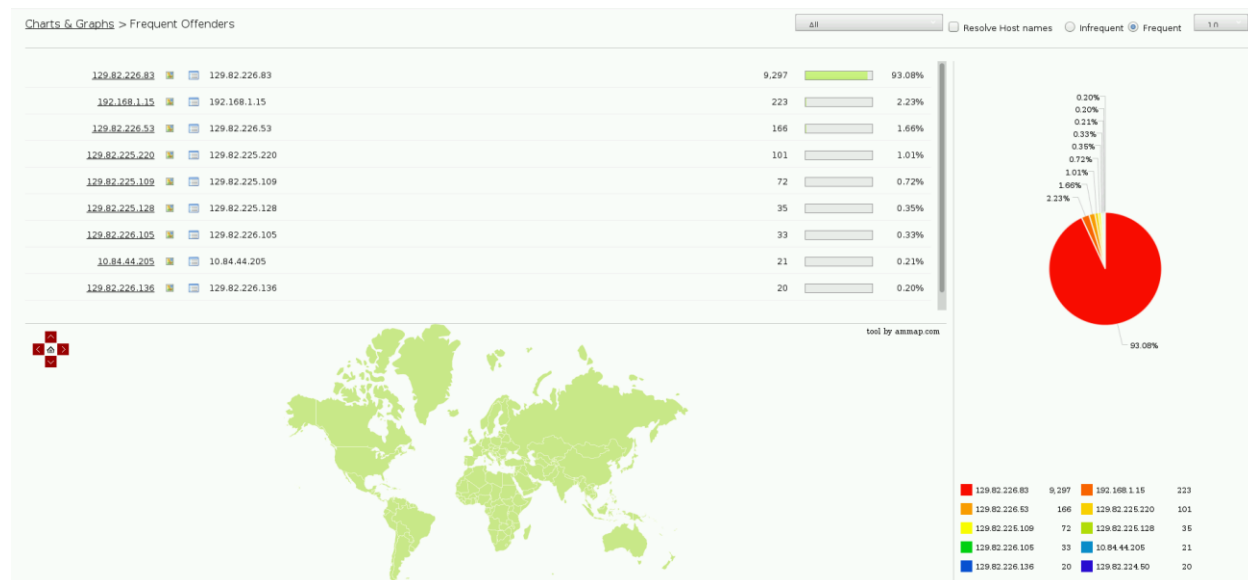


Figure 41: Aanval's Frequent Offender Page

Future Recommendations for Using Aanval

The Cyber Security Team secured the SAS Enterprise License fairly late in the semester, after the Cyber Cerburi dashboard interface had already been completed. The team recognizes that Aanval's interface is more powerful than what has been created for the Cyber Cerburi dashboard and detail pages, and would recommend that future work continue by improving the functionality of the set-up wizard and focusing on utilizing Aanval as the primary Snort web interface.

The team recommends that the current Cyber Cerburi dashboard and detail pages should be restructured to focus on analyzing data made available by the Kippo honeypot software.

Database Integration for Rendering Charts

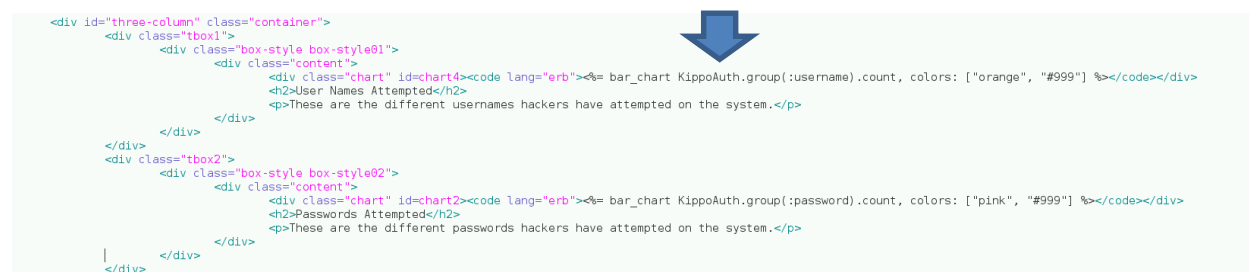
Each chart that was implemented in the dashboard was created with the "googlecharts" and "chartkick" gems, and is created when an array of information is passed to it. This information is in the form of a Rails model that is created dynamically when the dashboard page is loaded. When the HTML page is loaded in a web browser, embedded Ruby commands call the dashboard's controller file. The controller file is where the command to dynamically supply the model is located.

The actual model is then built when the controller executes this model initialization (by creating a variable in which to store the model information). As this initialization command requires code located in the model's individual Ruby file to be executed.

Each model required for a chart utilizes an "Active Record" to poll and search the Kippo and Snort database tables for updated entries. Any new entries found in these tables are then used to build an information entry for the associated model (this is basically a new entry for the model's database table). This updated information is then supplied (in the form of the model) to the gem, allowing a chart to be rendered on the dashboard form.

Utilizing the “Active Record” functionality to dynamically build the contents of the cerburi database prevents the need to run Rail’s seed commands, as well as maximizes the limited operating performance of the Raspberry Pi’s hard drive by only “moving” new information generated by Kippo and Snort when a user requests to view this information.

Figure 42: Embedded Ruby Code in the HTML Page displays an example of the embedded Ruby code located inside the dashboard page responsible for rendering a chart. While *Figure 43: Initialization of the Model* displays the initialization command for the model located inside of the dashboard’s controller file, and *Figure 44: Creating the Active Base Record* shows how the “Active Record” functionality is used to dynamically build the model supplied.

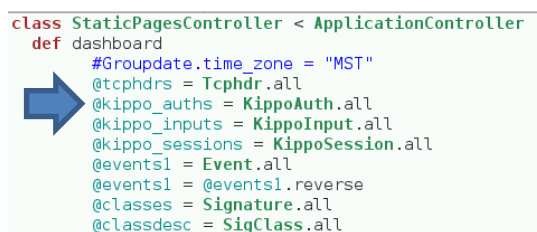


```

<div id="three-column" class="container">
  <div class="tbody">
    <div class="box-style box-style01">
      <div class="content">
        <div class="chart" id=chart4><code lang="erb"><%= bar_chart KippoAuth.group(:username).count, colors: ["orange", "#999"] %></code></div>
        <h2>User Names Attempted</h2>
        <p>These are the different usernames hackers have attempted on the system.</p>
      </div>
    </div>
    <div class="tbody">
      <div class="box-style box-style02">
        <div class="content">
          <div class="chart" id=chart2><code lang="erb"><%= bar_chart KippoAuth.group(:password).count, colors: ["pink", "#999"] %></code></div>
          <h2>Passwords Attempted</h2>
          <p>These are the different passwords hackers have attempted on the system.</p>
        </div>
      </div>
    </div>
  </div>
</div>

```

Figure 42: Embedded Ruby Code in the HTML Page



```

class StaticPagesController < ApplicationController
  def dashboard
    #Groupdate.time_zone = "MST"
    @tcphdrs = TcpHdr.all
    @kippo_auths = KippoAuth.all
    @kippo_inputs = KippoInput.all
    @kippo_sessions = KippoSession.all
    @events1 = Event.all
    @events1 = @events1.reverse
    @classes = Signature.all
    @classdesc = SigClass.all
  end
end

```

Figure 43: Initialization of the Model

```

1 class KippoAuth < ActiveRecord::Base
2 begin
3     @kippoauths = KippoAuth.all
4
5     con = Mysql.new 'localhost', 'root', 'honey', 'kippo'
6
7     #load_auth
8     lastentry = DateTime.new(2008, 8, 29, 0, 0, 0)
9     lastentry2 = DateTime.new
10
11     rs = con.query "SELECT * FROM auth"
12     rs.each_hash do |row|
13         if row['timestamp']>lastentry
14             lastentry = row['timestamp']
15         end
16     end
17
18     #get the last items only
19     @kippoauths.each do |row|
20         lastentry2 = row['time_stamp']
21     end
22     if @kippoauths.count == 0
23         lastentry2 = DateTime.new(2008, 8, 29, 0, 0, 0)
24     end
25     if lastentry2 < lastentry
26         rs = con.query "SELECT * FROM auth where timestamp>='"+lastentry2.to_s+"'"
27         rs.each_hash do |row|
28             session_con = row['session'].to_s
29             suc_con = row['success'].to_i
30             user_con = row['username'].to_s
31             pass_con = row['password'].to_s
32             time_con = row['timestamp']
33             KippoAuth.create session:session_con, success:suc_con, username:user_con, password:pass_con, time_stamp:time_con
34         end
35     end
36 end
37 ensure
38 con.close if con
39 end
40 end

```

Figure 44: Creating the Active Base Record

Proof of Concept: Testing the Cyber Cerburi System

The Cyber Cerburi system was tested in-house to gauge the functionality of the third-party software installed, as well as the stability of the graphical user interface that was developed. The aim of these tests was to provide a proof of concept for the Cyber Cerburi system, in its ability to detect and report findings associated with malicious network activity.

Testing Configuration

In order to successfully conduct a network penetration test, the team ensured that the Kippo honeypot, the Rails webserver, Apache2, Snort, Barnyard2, and Aanval's processors were all up and running.

The Cyber Cerburi system has been equipped with a batch script, titled "SnortBarn" that automates the initialization of these programs at startup. This batch script is located in the /etc/init.d directory.

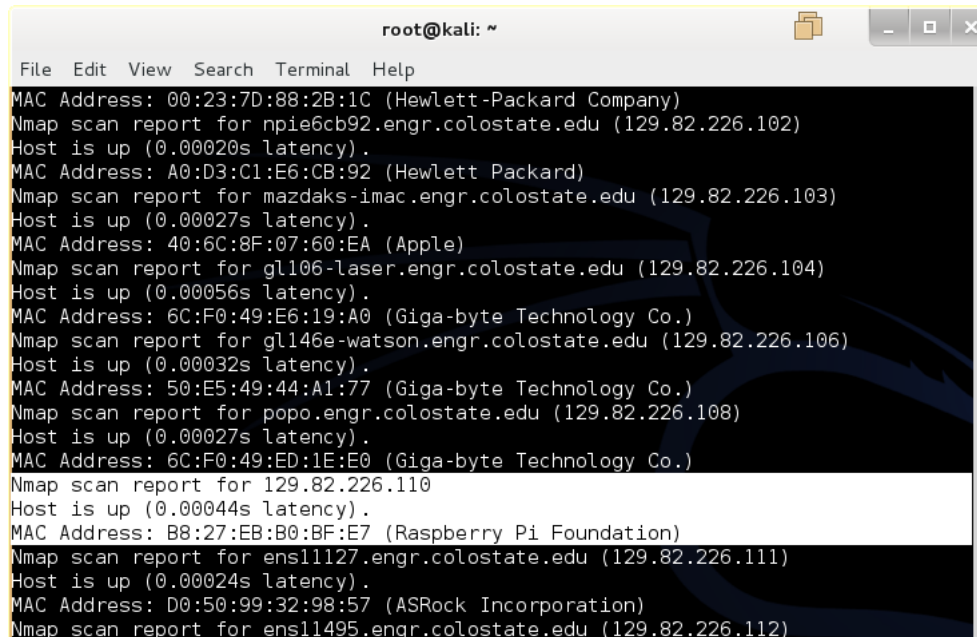
The computer used to carry out the following attacks was a combination of the Kali Linux x64 virtual machine, as well as the Ubuntu development virtual machine. Both machines were hosted by Cyber Security Team's desktop computer, which is located on the same physical network as the Cyber Cerburi system.

Port Scanning

Ports scans are typically conducted utilizing the Linux-based "nmap" tool to provide network reconnaissance information. When an attacker scans ports, their usual intent is to find out which ports are open on an intended victim's computer in order to select an appropriate exploit. The process of selecting an exploit that will successfully breach a system is closely tied to the port (and thus type of protocol) that can be manipulated.

Procedure

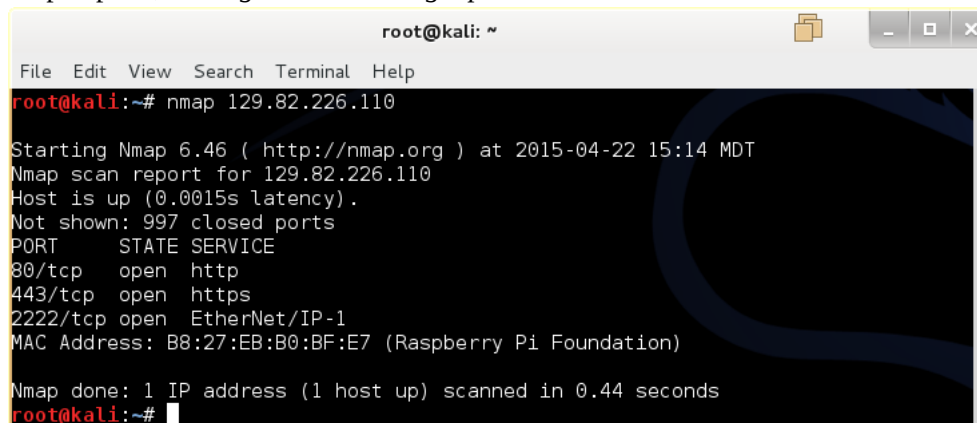
1. The nmap sweep was conducted from the attack computer by entering “nmap -sP 129.82.226.0/24” in order to scan the available devices on the network.
2. The Cyber Cerburi system was located on the network through the list provided by this search, and its IP address was noted (see *Figure 45: Locating the Cyber Cerburi System on a Network* below).



```
root@kali: ~  
File Edit View Search Terminal Help  
MAC Address: 00:23:7D:88:2B:1C (Hewlett-Packard Company)  
Nmap scan report for npie6cb92.engr.colostate.edu (129.82.226.102)  
Host is up (0.00020s latency).  
MAC Address: A0:D3:C1:E6:CB:92 (Hewlett Packard)  
Nmap scan report for mazdaks-imac.engr.colostate.edu (129.82.226.103)  
Host is up (0.00027s latency).  
MAC Address: 40:6C:8F:07:60:EA (Apple)  
Nmap scan report for gl106-laser.engr.colostate.edu (129.82.226.104)  
Host is up (0.00056s latency).  
MAC Address: 6C:F0:49:E6:19:A0 (Giga-byte Technology Co.)  
Nmap scan report for gl146e-watson.engr.colostate.edu (129.82.226.106)  
Host is up (0.00032s latency).  
MAC Address: 50:E5:49:44:A1:77 (Giga-byte Technology Co.)  
Nmap scan report for popo.engr.colostate.edu (129.82.226.108)  
Host is up (0.00027s latency).  
MAC Address: 6C:F0:49:ED:1E:E0 (Giga-byte Technology Co.)  
Nmap scan report for 129.82.226.110  
Host is up (0.00044s latency).  
MAC Address: B8:27:EB:B0:BF:E7 (Raspberry Pi Foundation)  
Nmap scan report for ens11127.engr.colostate.edu (129.82.226.111)  
Host is up (0.00024s latency).  
MAC Address: D0:50:99:32:98:57 (ASRock Incorporation)  
Nmap scan report for ens11495.engr.colostate.edu (129.82.226.112)
```

Figure 45: Locating the Cyber Cerburi System on a Network

3. With the IP address of the Cyber Cerburi now known, an nmap scan was performed on the system to locate open ports, see *Figure 46: Finding Open Ports*.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nmap 129.82.226.110  
Starting Nmap 6.46 ( http://nmap.org ) at 2015-04-22 15:14 MDT  
Nmap scan report for 129.82.226.110  
Host is up (0.0015s latency).  
Not shown: 997 closed ports  
PORT      STATE SERVICE  
80/tcp    open  http  
443/tcp   open  https  
2222/tcp  open  Ethernet/IP-1  
MAC Address: B8:27:EB:B0:BF:E7 (Raspberry Pi Foundation)  
Nmap done: 1 IP address (1 host up) scanned in 0.44 seconds  
root@kali:~#
```

Figure 46: Finding Open Ports

Results & Discussion

The Cyber Cerburi and Aanval interface both display the activity captured by Snort that was stored in the database by listing the type of threat detected, its description, priority level, and time stamp of occurrence.

98	11242	2	portscan: TCP Portscan	attempted-recon	221	Apr 23, 2015, 5:01:31 PM
----	-------	---	------------------------	-----------------	-----	--------------------------

Figure 47: Recorded Port Scanning Activity by Cyber Cerburi

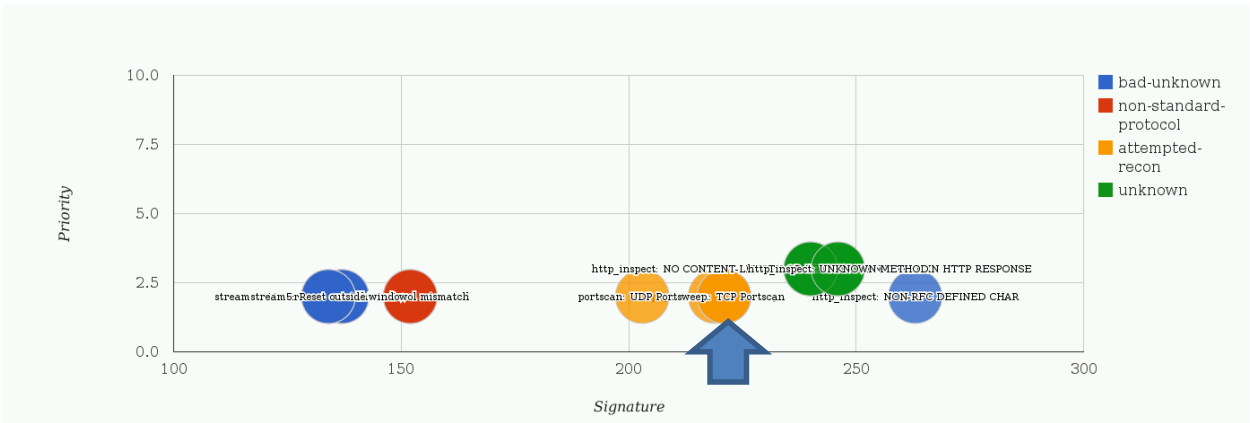


Figure 48: Recorded Port Scanning Activity by Cyber Cerburi

2	portscan: TCP Portscan	Cerburi	129.82.21.88	255
2	attempted-recon	Cerburi	129.82.225.132	
2	ssh: Protocol mismatch	Cerburi	129.82.225.86	52148
2	non-standard-protocol	Cerburi	129.82.224.66	22 TCP
2	ssh: Protocol mismatch	Cerburi	129.82.225.86	52148
2	non-standard-protocol	Cerburi	129.82.224.66	22 TCP

Figure 49: Recorded Port Scanning Activity by Aanval

As seen in Figure 49: Recorded Port Scanning Activity by Aanval, a more thorough evaluation of the Snort data has been provided – such as the IP address from where the attack originates. In addition, Aanval can summarize other important information captured by Snort in a small report. The port scan will be listed as “recon” in the report.

Nmap can be used to detect how many possible victims are located on a given network by providing available IP addresses. Once a target machine is chosen, Nmap can then speculate as to what type of operating system the machine is running, as well as specify information regarding currently opened ports.

Having the ability to detect this type of activity is pertinent in possibly preventing a future attack, and as demonstrated by the test above, the Cyber Cerburi system does have the ability to record and report this information to users.

Brute Force Attacks

Cyber Cerburi has been set up to handle brute force attacks utilizing the Kippo honeypot. Kippo is designed to simulate a semi-vulnerable machine on a network that can be accessed through SSH protocol. By default, Kippo is set to create an imitation file system (deemed “fs.pickle”) that is generated from the system it is installed on. This imitation file system, however, can be changed to mimic a multitude of other operating and file system types.

For testing purposes, the Kippo imitation file system created was based upon the initial Raspbian distribution included with the Raspberry Pi module. Additionally, a hostname, password and port number

can be set for the imitation file system. Kippo saves all information regarding brute-force attacks in the Kippo MySQL database – titled “kippo”. The following settings were established for testing:

IP: localhost, this varies as the IP is assigned by the CSU network’s DNS (Domain Name Server).

Port: 2222

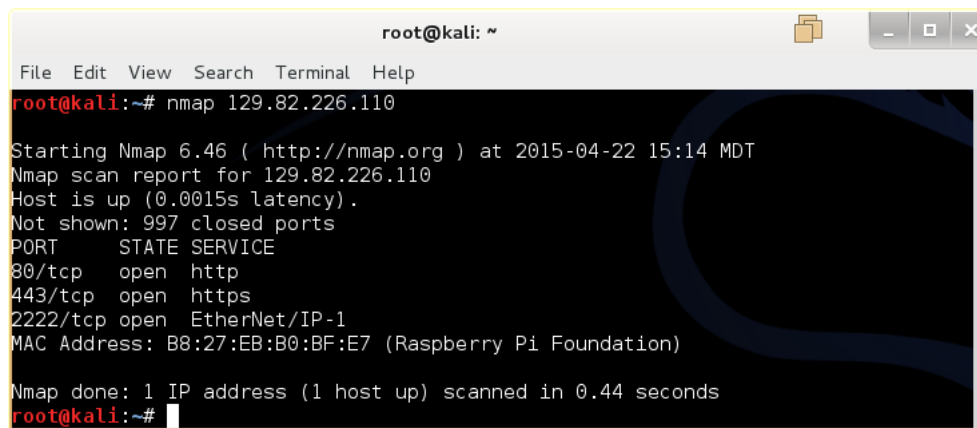
Hostname: nas3

Password: 123456

These settings can be changed within the Kippo configuration file located at /usr/sr/kippo-0.8/kippo.cfg.

Procedure

1. An Nmap scan was executed from the attack computer to locate any open ports on the honeypot (IP address of the honeypot was already known). As seen in the screenshot below (*Figure 50: Nmap Scan of the Cyber Cerburi System*) port 2222 has been opened by the Kippo honeypot.



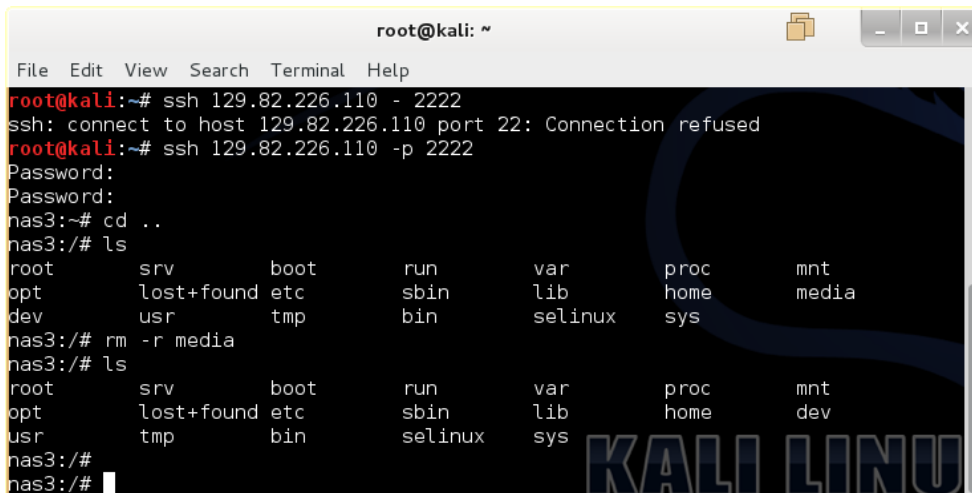
```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nmap 129.82.226.110

Starting Nmap 6.46 ( http://nmap.org ) at 2015-04-22 15:14 MDT
Nmap scan report for 129.82.226.110
Host is up (0.0015s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
2222/tcp  open  Ethernet/IP-1
MAC Address: B8:27:EB:B0:BF:E7 (Raspberry Pi Foundation)

Nmap done: 1 IP address (1 host up) scanned in 0.44 seconds
root@kali:~#
```

Figure 50: Nmap Scan of the Cyber Cerburi System

2. Entering the SSH command from the attack computer with the specified vulnerable port prompts for a password to be entered. If the password entered is wrong, Kippo will continue to ask for a password until the correct password is entered. Kippo simulates a real system by reporting output like a standard shell would. This functionality allows a hacker to believe that they have successfully breached the targeted machine and now has full access to the file system (see *Figure 51: Brute Force Attack* below).



```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# ssh 129.82.226.110 - 2222
ssh: connect to host 129.82.226.110 port 22: Connection refused
root@kali:~# ssh 129.82.226.110 -p 2222
Password:
Password:
nas3:~# cd ..
nas3:/# ls
root      srv      boot    run      var      proc     mnt
opt      lost+found etc     sbin     lib      home     media
dev      usr      tmp     bin      selinux  sys
nas3:/# rm -r media
nas3:/# ls
root      srv      boot    run      var      proc     mnt
opt      lost+found etc     sbin     lib      home     dev
usr      tmp     bin      selinux  sys
nas3:/#
nas3:/#
```

Figure 51: Brute Force Attack

3. A “wget” command was performed from the attack machine to download a malicious file to the target machines file system (see *Figure 52: Performing a "wget" Command during a Brute-Force Attack* below).



```
nas3:~# wget http://upload.wikimedia.org/commons/2/24/Male_mallard_duck_2.jpg
--2015-04-27 13:20:53-- http://upload.wikimedia.org/commons/2/24/Male_mallard_duck_2.jpg
Connecting to upload.wikimedia.org:80... connected.
HTTP request sent, awaiting response... 404 Not Found
nas3:~# wget http://upload.wikimedia.org/wikipedia/commons/2/24/Male_mallard_duck_2.jpg
--2015-04-27 13:23:59-- http://upload.wikimedia.org/wikipedia/commons/2/24/Male_mallard_duck_2.jpg
Connecting to upload.wikimedia.org:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1800492 (1M) [image/jpeg]
Saving to: `Male_mallard_duck_2.jpg'

100%[=====] 1,800,492 56K/s eta 31s

2015-04-27 13:24:00 (56 KB/s) - `Male_mallard_duck_2.jpg' saved [1800492/1800492]
nas3:~#
```

Figure 52: Performing a "wget" Command during a Brute-Force Attack

Results & Discussion

The Kippo honeypot automatically saves any downloaded files to the malware folder, titled “dl”, which acts as a vault to prevent execution of the files on the operating system. *Figure 53: Capturing and Storing Downloaded Malware* displays the contents of the “dl” folder after the “wget” command was conducted.

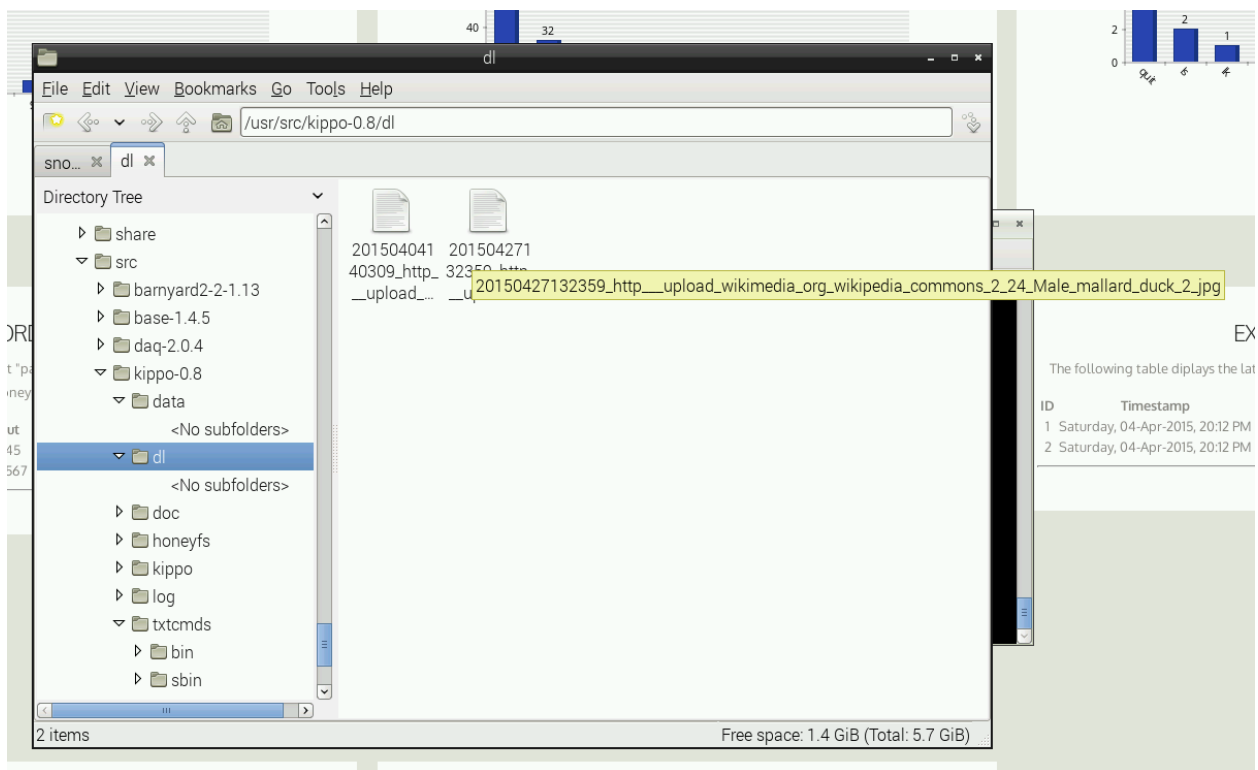


Figure 53: Capturing and Storing Downloaded Malware

The “wget” command initiated during the brute-force attack is also displayed on the Kippo-Graph web interface, as see below in Figure 54: "wget" Command Recorded by Kippo.

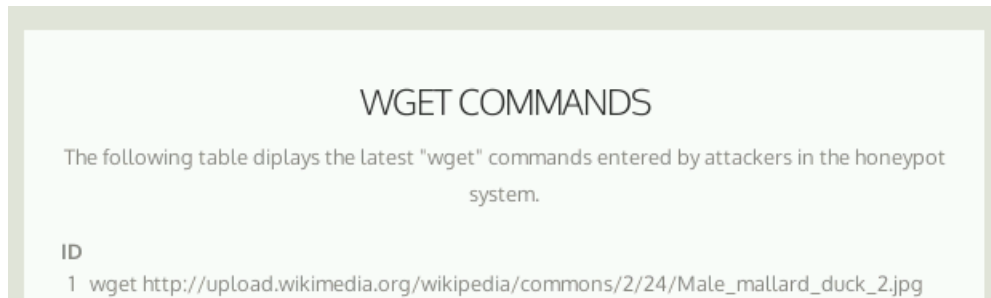


Figure 54: "wget" Command Recorded by Kippo

The geolocation of the attacker was also narrowed down to a city and state, as determined through the use of a WHOIS database query (see Figure 55: Geolocation Information Provided below).

ID	IP Address	Probes	City	Region	Country Name	Code	Latitude	Longitude	Hostname	Lookup
1	129.82.226.83	32	Fort Collins	CO	United States	US	40.455502	-105.464798	brimstone_2.engr.colostate.edu	
2	129.82.226.53	18	Fort Collins	CO	United States	US	40.455502	-105.464798	129.82.226.53	

Figure 55: Geolocation Information Provided

The top brute-force attack sources are also listed in the Kippo-Graph web interface, as seen below in Figure 56: The Top SSH Clients Recorded by Kippo (Available in Kippo-Graphs).

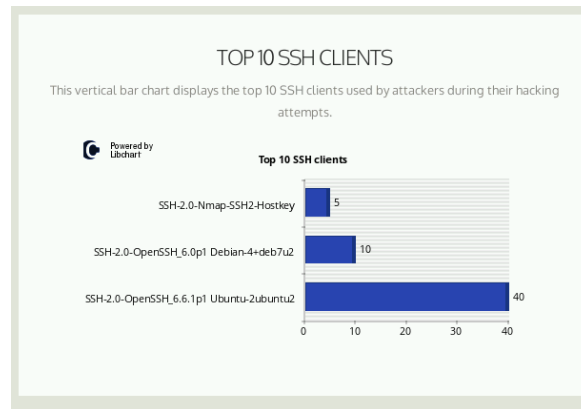


Figure 56: The Top SSH Clients Recorded by Kippo (Available in Kippo-Graphs)

The Cyber Cerburi interface also kept tabs on the brute-force attack, and lists the username as well as the command line input attempted (see *Figure 57: Commonly Used Usernames (Reported by Cyber Cerburi)* and *Figure 58: Command Line Hacking Inputs (Recorded by Cyber Cerburi)* below).

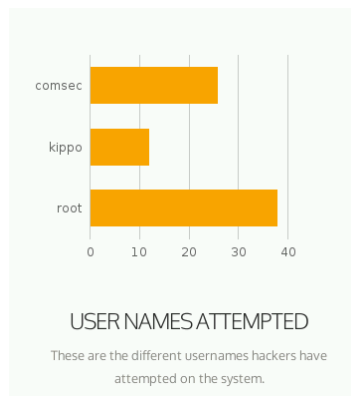


Figure 57: Commonly Used Usernames (Reported by Cyber Cerburi)

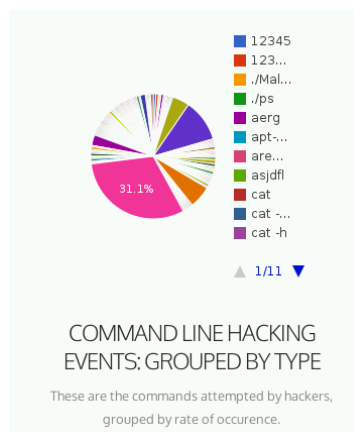


Figure 58: Command Line Hacking Inputs (Recorded by Cyber Cerburi)

Kippo has proven to be a useful medium interaction honeypot that is capable of deflecting brute force attacks. The team has concluded that the honeypot is working successfully, as Kippo has the ability to reset the fake file system after each SSH connection that is made (in addition to reporting detailed information about the hacker's actions).

Proof of Concept: Conclusion

Port scans can be an indication that an attack is about to occur. An attacker will typically try to exploit a target computer through one of the established opened ports that is found during a network port scan.

When running the attack from the Kali Linux virtual machine, the scan was recorded and captured by the Snort intrusion detection software located on Cyber Cerburi's system. However, not only was the attack captured, but other useful information (such as the attacker's IP address and the port number targeted) was also captured and reported. Although port scanning is just one aspect of the types of activity Snort is capable of capturing, it is one of the most important as it is almost always the preliminary move of a hacker before executing an attack. The success of the team's proof of concept testing is also indicative that the intrusion detection system equipped on Cyber Cerburi is fully functional.

Brute force attacks are a quick and easy way in which an attacker can gain access to a computer without having to go to the trouble of implementing an exploit. Having access to the recorded movements of an attacker is key to understanding what their motives may have been.

Utilizing the Kippo honeypot as a vulnerable computer has proven to be a very useful tool in gathering this type of information. When a brute force attack was carried out, the attacker was able execute commands provided by a typical shell interface. The functionality to move, copy, delete and download files indicates to an attacker that they have complete control over a system. However, instead of manipulating actual system files, file manipulation occurred within the honeypot's virtual file system, and was reset once the brute-force attack session had ended.

Evidence of the brute-force attack was provided by Kippo successfully logging all of the hacker's command line activity. The Kippo-Graphs interface, along with Cyber Cerburi's graphical user interface, provided a visual representation of the evidence collected. Overall, the Cyber Cerburi system has proven to be a device fit for deflecting brute force attacks -- both through the use of controlled manipulation of the hacker's actions, as well as through verifiably reporting events that occurred to the user.

The Cyber Cerburi System vs. Window's Honeypots

The Cyber Cerburi system possesses significant improvements over the Windows-based honeypot software: Honeybot and KFSensor. Honeybot, the first free honeypot software program utilized by the team, was essentially no more than an intrusion detection system. However, Honeybot was very limited in its "data-packet sniffing" functionality and rarely captured network activity (this lapse in reliable behavior was attributed to bugs in the software program itself).

Therefore, it was not long before the team switched over to utilizing KFSensor's honeypot software. A professional license costs \$599.00, but the team was able to obtain an educational license for a limited-time frame. While KFSensor proved to be more reliable than Honeybot in terms of intrusion detection capabilities, it still had many issues.

The first issue encountered by the team was that KFSensor still failed to detect a majority of the the exploitive and port scanning activity during in-house testing. The second issue encountered was that KFSensor also failed to create log files that were crucially needed to analyze data. Finally, the third issue encountered was that KFSensor promised the ability to have service emulation. However, since this feature is considered as an "advanced" user option, there is very little documentation available on how to correctly implement it, and even after numerous troubleshooting efforts the team was never able to get this feature implemented correctly. Eventually the decision was made to use a free version of Wireshark

as an intrusion detection system for data analysis. However, Wireshark captures all activity on the network, and although filters can be implemented, data-packets are still only filtered by protocol type (UDP/TCP). Security definitions are not implemented in Wireshark, therefore the team had to determine what was (and what was not) malicious activity on a packet by packet basis – which was both time-consuming and highly ineffective.

The movement from Windows to a Linux platform when developing the Cyber Cerburi system was one of the best choices the team could have made, as the Snort intrusion detection system not only reliably detected malicious activity on the network, but also implemented highly accurate security definitions for data-packet sniffing (through the use of the PulledPork software add-on). Therefore, non-malicious network activity was not reported, and the events that were reported as malicious were classified by their threat level.

The Cyber Cerburi system also increased its honeypot functionality by utilizing Kippo's honeypot software package. Kippo enabled the use of an SSH remote shell, as well as the ability to emulate a fake file system that attracts hackers to the honeypot. Kippo also added the ability to collect potential malware that the attacker downloaded during an exploit and stored it in a secure location on the Cyber Cerburi system.

Finally, the easy to use set-up interface created by the Cyber Security Team greatly simplified the process of using Snort, as well as Kippo, and was targeted towards non-technical users. The set-up wizard modified necessary configuration files needed to make the Cyber Cerburi system operable on a network. In addition, once the set-up process was complete, the dashboard provided a convenient and easy-to-understand graphical user interface for viewing the data collected by Snort and Kippo in real-time.

Known Issues & Recommendations for System Improvement

Overall the team's efforts this semester have proven fruitful in rendering a functional, portable module that provides both intrusion detection and honeypot capabilities to deter malicious activity. However, looking back over the various efforts implemented throughout the development of this system, there are naturally some known issues as well as recommendations for future teams that need to be disclosed.

Issue	Description	Recommendation
Administrative Access for the Cyber Cerburi System	Currently there is no "Administrative Access" page available for the Cyber Cerburi system. Therefore, once a user and password is created there is no way for either of these to be altered and/or deleted except by accessing the MySQL database directly.	This page needs to be developed to allow users to both modify and/or recover a password if needed. In addition, the ability to add filters to Snort and Kippo functionality should be incorporated.
Oink-Code Registration is Not Optional	Users are not allowed to opt-out of registration for the oink-code, as it is a required step in the installation process.	Default security definitions should come preloaded with the system if the user does not wish to register for an oink-code. In addition, this should be an optional step in the installation process for the user.
Auto Updates for PulledPork Not Yet Implemented	Auto updating functionality for PulledPork has not been added. In the past these updates were completed by manually entering the update command in the terminal.	Auto update configuration should be completed utilizing PulledPork's "crontab.log" option. The time/frequency of these updates should be pulled from settings set by the user in the future "Administrative Access" page.
Duplicate Usernames are Allowed	Currently the Set-up Wizard allows duplicate usernames to be entered.	This is a fairly simple fix, which could be incorporated by making the "username" field in the "user" database table a primary key (as primary keys have to be individual, non-null values).
Ruby on Rails Architecture Choice	Ruby on Rails requires its own database to work properly. However, as Snort and Kippo already supply databases, this has led to unnecessary data transfer between databases.	If Ruby on Rails is elected for use in the continuation project, the Cerburi database table should only hold refined information from Snort & Kippo (which can lead to better analysis of the information returned). Only refined information should then be stored in the "cerburi" database to minimize the existence of duplicate data.

Issue	Description	Recommendation
Internet Access is Required to Utilize the Chartkick and GoogleCharts Gems	The Cyber Cerburi system must maintain access to the internet in order to render both the Chartkick and GoogleCharts charts displayed in the web interface.	Different gems should be selected for use to generate charts that do not require “active” checkouts in order to be rendered on the screen.
Apache2 & Ruby on Rails Webservers	Utilizing two webserver places a large workload on the Raspberry Pi’s processor.	Future groups should consider either moving towards an interface that requires an Apache2 webserver to support it, or completely eliminating the need for Kippo-Graphs.
Active Back-Up and Roll-Over of Database Tables Not Implemented	Currently the Snort, Kippo, and Cyber Cerburi databases are not limited in the amount of information they can contain, which can lead to memory shortage problems.	Functionality should be added to limit the amount of active information contained in all of the databases, and older information should either be deleted, or moved to an external location for storage.
Ability to Alert User When Certain Malicious Activity Occurs on the Network Not Yet Implemented	The Cyber Cerburi system does not currently alert users if a brute force attack is currently underway.	Functionality should be added to obtain an email address from the user in which automated emails can be sent from Cyber Cerburi’s system to give active notifications, and possibly directions to follow, in the event that malicious activity is suspected.

Table 24: Known Issues & Recommendations for System Improvement

Future Development Board Recommendations

There are several directions the future cyber security team can take the project. The two main directions are a hacking/penetration testing approach and/or continuation of the small business based honeypot (Cyber Cerburi) approach.

An informative tutorial, titled [Penetration Testing with Raspberry Pi](#), was recommended by Dr. Dubow and Dr. Collins. This tutorial provides detailed steps on how to develop a disguised attack Raspberry Pi that is loaded with an ARM version of Kali Linux.

The basis of this idea is to get a Raspberry Pi in proximity of a target network (which should be established by the Cyber Security Team for legal and ethical reasons). The Raspberry Pi is accessed through a 3G/4G connection by the attacker and executes wireless (WIFI) cracking software. Once the WIFI password is obtained, the attacker can then access any computer inside the network and have full shell control once systems were breached.

If the second approach (to continue development of the Cyber Cerburi system) is chosen, a different development board should be chosen by the team, as the Raspberry Pi has continued to face issues related its limited operating performance.

Besides the items listed below in *Table 25: Development Board Pros & Cons*, there were two other devices that were considered. These were the Chromebit and Intel's computer stick. Although these are both powerful devices, they lacked the number of peripherals required to make a robust honeypot, namely the Ethernet adapter.

Not having an Ethernet adapter is a huge issue when using Snort as an intrusion detection system, because Snort does not support data packet capturing of wireless network traffic.

The team recommends the following devices for use as a development board. However, it is strongly advised to use this information as a basis for further research as newer, more powerful boards may be released since the time of this writing.

Device / Price	Pros	Cons
Gizmo 2 / \$200.00	• USB 3.0 • 1 GHz 32-bit CPU (significant advantages over ARM CPU) • 1 GB DDR3 RAM (speed being the only advantage) • MSATA for connecting external drives with fast data transfer rates • Linux and Windows supported	Cost is significantly higher (Pi 2 is around \$35), compared to \$200 for not much more performance
MIPS Creator C120 / \$65.00	• 1.2 GHz 32-bit CPU • 8 GB Flash Drive • WIFI capable • Bluetooth capable • 1 GB DDR3 RAM • Debian Linux OS	Does not provide much more computing power than the Raspberry Pi 2
ODROID-XU3 Lite / \$99.00	• 1.8 GHz Quad Core ARM Processor • 2GB DDR3 RAM • USB 3.0 • Expandable SSD Memory Modules	Higher cost than Raspberry Pi 2, has an ARM processor
ODROID-XU3 / \$179.00	• 2.0 GHz Quad Core ARM Processor • 2GB DDR3 RAM • USB 3.0 • Internal Storage 16/32/64 GB in addition to the Micro-SD slot • Ubuntu 14.04 OS and other Linux supported •	Higher cost than Raspberry Pi 2, as the 32GB module starts at \$179.00, also has an ARM processor

Device / Price	Pros	Cons
Mint Box / \$295.00	• Quad Core 1.0 – 1.6 GHz processor • 4GB DDR3 RAM • 64GB mSATA Solid State Drive • Linux Mint supported	Higher cost than Raspberry Pi 2, as the 1.0 GHz processor starts at \$295.00

Table 25: Development Board Pros & Cons

Out of all of the boards listed above, the Odroid UX3 Lite development board is the highest recommended due to its attractive price and increased performance over the Raspberry Pi 2. Additionally, it can use a full version of Ubuntu 14.14 which should encounter very little limitations when installing software.

If this project continues into next year, two of these boards will be ordered and available for next year's team. A development board that can support Windows would be an easier route when it comes to setting up the software. However, the software available for Windows is often costly and not capable of performing as many tasks as similar software packages available for Linux.

Conclusion

Computer security will continue to be an issue for years to come. However, with tools such as intrusion detection systems and honeypots, countering the threat of attackers should be a manageable solution. The Cyber Security Team at Colorado State University hoped to add to the cyber security solution by building a powerful desktop computer capable of running multiple virtual machines at once. The goal of this computer was to simulate live networks currently being used to learn more about cyber security.

In addition, the team has created a portable honeypot solution. The target platform for this was the Raspberry Pi 2, chosen for its small form-factor, low cost, and decent processing power. A multitude of software had to be installed on the Raspberry Pi 2 to accommodate for data visualization and proper statistical-gathering of the honeypot and intrusion detection system.

To enhance the project, future teams should add more honeypots to the solution and provide more user functionality in the Cyber Cerburi interface on a different development board due to limitations presented by the Raspberry Pi 2. Finally, our team recommends continuing to utilize Aanval as the primary front-end for Snort, while continuing to develop a more comprehensive front-end for the Kippo honeypot (as an alternative to Kippo-Graphs).

In conclusion, network security solution products aimed at small businesses have a lot of potential value in the today's consumer market. Developing an affordable, portable, easy-to-use device offers many small businesses a way in which to safeguard their online security. Extending the usability and functionality of the device are critical steps in making this product usable, and once these steps are completed, this product will be a highly valuable cyber security tool.

References

[1] National Cyber Security Alliance, *America's Small Businesses Must Take Online Security More Seriously*, [online], Available: <https://www.staysafeonline.org/stay-safe-online/resources/small-business-online-security-infographic>, Accessed: April 26, 2015.

Appendix A: In-House Testing Exploring Honeypots: An Instruction into Ethical Hacking and Intrusion Detection: Findings & Recommendations

About the Honeypot Lab

The entire honeypot lab manual was reread and completed in-house to ensure that it will be as clear and concise as possible for future students who elect to work with it. Overall, the instruction manual remains a valuable resource for introducing many topics associated with network penetration testing and honeypots.

The manual details basic information and case studies involving honeypots, intrusion detection systems, and ethical hacking techniques. The manual then divulges the details of steps required to set up a network penetration testing environment, conduct an exploit, install a backdoor, as well as perform forensic analysis with Autopsy. The appendix also contains supplementary information involving various topics, with a focus on topics related to network penetration testing and troubleshooting.

Findings

After a thorough review of each section, it was found that minimal changes had to be made. The bulk of the changes had to do with unclear steps, or commands that did not yield expected results. However, with the exception of only one command that needed to be altered, the bulk of these changes were easy to make due to the predominant use of linked (dynamic) formatting throughout the document.

In a few areas of the paper, specifically the “Exploit” and “Backdoor” sections, additional material had to be added in order for the expected results to be achieved. Finally, the paper concludes with possible goals about future tasks related to cyber security for the team. While these ideas are good, the actual idea used for second semester was quite different than what was discussed in the conclusion of lab. Whether or not this section needs to be revised in order to reflect the actions actually followed by the team is questionable at this time.

Recommendations

The manual could be improved by providing additional material in the “Network Reconnaissance” stage, as well as steps on how to target additional computers inside of a network once one computer was breached. Even small modifications to the manual highlight the fact that cyber security is a rapidly changing field and that specific exploits may not work in the future. However, with that being said, the amount of fundamental knowledge that this instruction manual presents will prove to be useful for many generations of interested students.

Additionally, the lab manual contains many details pertaining to network penetration testing in one, compact document. This type of document, which contains many fundamental topics of cyber security, is not easily found in the academic realm; this alone establishes the instruction manual as a useful and valuable document.

Appendix B: Abbreviations

A list of abbreviations used throughout the manual are defined below:

- DDR2 – Double Data Rate 2
- DDR3 – Double Data Rate 3
- DoS – Denial of Service
- DDoS – Distributed Denial of Service
- GB – Gigabyte
- GHz – Gigahertz
- GUI – Graphical User Interface
- HDD – Hard Disk Drive
- IDS – Intrusion Detection System
- IP – Internet Protocol
- ISO – Uncompressed Disc Image
- NAT – Network Address Translation
- OS – Operating System
- RAM – Random Access Memory
- TCP – Transmission Control Protocol
- USB – Universal Serial Bus
- VM – Virtual Machine
- VPN – Virtual Private Network

Appendix C: Summary of Team Spending

The Cyber Security Team was awarded \$2,400.00 in funding by the Engineering Student Technology Committee for creation of a Cyber Security Training Lab on campus at Colorado State University. At this time, the allotment for each piece of equipment in the Training Lab has been determined, and the updated project budget is enumerated below in Table 26: Finalized Spending Report.

Cyber Security Team 2014-2015 Finalized Spending Report					
Item	Description	Price	Qty.	Withdrawal	Remaining
Engineering Student Technology Committee Funding					\$1,800.00
Wireless Adapter	802.11b/g/n USB 2.0 Up to 150Mbps Wireless Data Rates	\$16.68/ea	1	\$16.68	\$1,783.32
Motherboard	ASUS M5A99X EVO R2.0 AM3+ AMD 990X	\$122.99/ea	1	\$122.99	\$1,660.33
Power Supply	CORSAIR CX series CX500 500W ATX12V	\$59.99/ea	1	\$59.99	\$1,600.34
Desktop Processor	AMD FX-8350 Vishera 8-Core 4.0GHz Socket AM3+	\$259.99/ea	1	\$259.99	\$1,340.35
Memory Model	G.SKILL Ripjaws X Series 32GB (4 x 8GB) 240-Pin DD	\$279.99/ea	1	\$279.99	\$1,060.36
Internal Hard Drive	WD BLACK SERIES WD1003FZEX 1TB 64MB Cache SATA 6.0Gb/s 3.5"	\$74.99/ea	1	\$74.99	\$985.37
HDMI-DVI Cable	HDMI CABLE COBOC EA-HD2DVI-6-BK R	\$3.19/ea	1	\$3.19	\$982.18
Shipping of Products	Shipment of Desktop PC Components from Newegg.com	\$6.77	1	\$6.77	\$975.41
Raspberry Pi Development Kits	Raspberry Pi 2 - Ultimate Starter Kit	\$109.95/ea \$14.00 Shipping	4	\$453.80	\$521.61
ODROID-XU3 Lite Development Boards	Development Boards for Future Pen Testing Kit	\$99.00/ea	2	\$198.00	\$323.61
32GB Linux Memory Module	Memory Module for Future Pen Testing Kit	\$49.00/ea	2	\$98.00	\$225.61
Additional Wireless Adapters	Wireless Adapters for Future Pen Testing Kit	\$8.00/ea	2	\$16.00	\$209.61
Micro USB 600mah Battery Pack	Battery Pack for Future Pen Testing Kit	\$21.99/ea	1	\$21.99	\$187.62
Unlocked USB Modem	USB Modem for Future Pen Testing Kit	\$21.92/ea	1	\$21.92	\$165.70
Mobile Hotspot	Mobile Hotspot Device for Future Pen Testing Kit	\$79.88/ea	1	\$79.88	\$85.82
Data Access Card	Data Access Card for Future Pen Testing Kit	\$25.00/ea	1	\$25.00	\$60.82
Estimated Shipping Costs	Estimated Shipping Costs for Materials to Construct the Future Pen Testing Kit	\$20.00/ea	1	\$20.00	\$40.82
Remaining Funds Available from Engineering Student Technology Committee					\$40.82
Electrical & Computer Engineering Department: Senior Design Team Funding					\$600.00
Shipping Overseas	Shipped a Raspberry Pi unit to a team member that was overseas	\$24.99/ea	1	\$24.99	\$575.01
USB Sticks	16 GB USB Memory Sticks	\$10.00/ea	4	\$40.00	\$535.01
T-shirts	Team T-shirts for Presentations	\$22.50/ea	4	\$90.00	\$445.01
Tablecloth	Tablecloth used for E-days booth	\$11.00/ea	1	\$11.00	\$434.01
Poster Board	Poster Board for E-days booth	\$4.00/ea	1	\$4.00	\$430.01
MicroSD Adapter	MicroSD Adapter used to Image Raspberry Pi	\$10.50/ea	1	\$10.50	\$419.51
Remaining Funds Available from Electrical & Computer Engineering Department					\$419.51

Table 26: Finalized Spending Report

Acknowledgements

The Cyber Security Team would like to express gratitude toward Dr. George Collins and Dr. Joel Dubow for the support and guidance they have provided throughout the year.

The knowledge they have contributed has gifted the Cyber Security Team with valuable motivation to learn more about the network security field and the ever constant battle to defend ourselves against black hat hackers. The lessons learned throughout the duration of project will prove beneficial for each team member in different, unique ways for many years to come.

Thank you for your time and attention; it was a pleasure working with both of you!

-The Cyber Security Team